⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平4−153733

⑤Int. Cl.[5]	識別記号	庁内整理番号
G 06 F 9/38	310 E	7927−5B
	310 X	7927−5B
	330 X	7927−5B
	370 B	7927−5B

⑬公開　平成4年(1992)5月27日

審査請求　未請求　請求項の数 1　(全36頁)

⑭発明の名称　並列処理装置における命令供給装置

⑳特　願　平2−279654
㉒出　願　平2(1990)10月17日

⑦発明者　安藤　秀樹　兵庫県伊丹市瑞原4丁目1番地　三菱電機株式会社エル・エス・アイ研究所内
⑦出願人　三菱電機株式会社　東京都千代田区丸の内2丁目2番3号
⑦代理人　弁理士　深見　久郎　外2名

明　細　書

1．発明の名称

並列処理装置における命令供給装置

2．特許請求の範囲

各々が所定の機能を実行する複数の機能ユニットと、命令を格納する命令メモリ装置と、命令フェッチ指示に応答して前記命令メモリ装置から複数の命令を同時にフェッチする命令フェッチ装置と、該フェッチされた複数の命令から同時に実行可能な命令を見出し、該同時実行可能な命令を関連の機能ユニットへ発行する命令デコード装置とを含む並列処理装置における命令供給装置であって、前記命令デコード装置は、前記関連の機能ユニットへ発行した命令数を示す情報と、与えられた命令に分岐命令が含まれておりこの分岐命令により分岐が発生したことを示す分岐発生指示情報とを発生する手段を含み、

前記命令メモリ装置からの命令の供給待ち状態であるか否かを示す命令供給有無指示信号と、リセット信号と、前記分岐発生指示情報とに応答して、命令の有効・無効を示す有効性フラグを形成する手段、

前記同時にフェッチされた複数の命令と、該同時にフェッチされた複数の命令の前記命令メモリ装置におけるアドレスと、前記有効性フラグとを同時に格納する記憶手段、前記記憶手段は、1エントリが1命令と対応のアドレスと対応の有効性フラグとからなる複数のエントリ格納領域を備えかつ、各々に異なる書込エントリが同時に伝達される複数の入力ポートおよび前記複数の入力ポートと別に設けられ、各々に異なる読出エントリが同時に伝達される複数の出力ポートと、書込キューアドレスおよび読出キューアドレスにそれぞれ応答して複数のエントリを前記複数の入力ポートおよび前記複数の出力ポートへ同時にそれぞれ接続する手段とを含み、および

前記命令発行数指示情報、前記分岐発生指示情報、前記命令供給有無指示信号、および前記リセット指示信号に応答して、前記書込キューアドレスおよび前記読出キューアドレスをそれぞれ生成

して前記記憶手段へ与えるアドレス生成手段を備え、

前記記憶手段から同時に読出された複数のエントリの内容が前記命令デコード装置へ与えられる、並列処理装置における命令供給装置。

3、発明の詳細な説明

[産業上の利用分野]

この発明は並列処理装置に関し、特に、スーパースカラー型プロセサの命令供給方式の効率化に関する。

[従来の技術]

近年のマイクロプロセサの進歩には目覚ましいものがあり、その高性能化とともに動作速度も高速化されてきている。一方、半導体メモリの高速化がこのマイクロプロセサの高速化に追随することができず、半導体メモリのアクセスがプロセサの高速化に対するボトルネックとなっている。このため、並列処理を行なうことによりマイクロプロセサの性能向上を図ることが行なわれている。

このような並列処理を実現する方式の1つに、スーパースカラーと呼ばれる処理方式がある。このスーパースカラー型のプロセサ（以下、単にスーパースカラーと称す）は、第30図に示すように、スーパースカラー内部のスケジューラ200が、命令ストリーム中の並列性を検出し、並列処理可能な命令を並列に設けられたパイプラインP1、P2およびP3へ供給する方式である。すなわち、スーパースカラーは以下の特徴をもつ処理装置であるといえる。

（1） 命令を複数個同時にフェッチする。

（2） 複数の機能ユニット（パイプライン）を有しており、複数の命令を同時に実行することができる。

（3） フェッチされた複数の命令の中から同時に実行可能な命令を見つけ出し、この同時に実行可能な命令を関連の機能ユニット（パイプライン）へ同時に投入する。

第31図はスーパースカラーの一般的構成を示す図である。第31図において、スーパースカラーは、各々が所定の機能を実行する複数の機能ユニット4．5．6および7と、命令を格納する命令メモリ1と、命令メモリ1から複数の命令を同時にフェッチする命令フェッチ（IF）ステージ2と、命令フェッチステージ2によりフェッチされた複数の命令を同時に受け、この中から同時に実行処理可能な命令を見出して対応の機能ユニットへ投入する命令デコード（ID）ステージ3と、演算処理結果等を格納するためのデータメモリ8とを含む。命令メモリ1は、キャッシュメモリおよび外部メモリを含み、キャッシュヒット（キャッシュメモリ内に要求されたメモリが存在する状態）の場合には高速で命令を読出すことができる。

命令フェッチステージ2は、命令メモリ2へ命令ポインタIF_PCを与え、この命令ポインタIF_PCに対応する複数の命令を命令メモリ1からフェッチする。

命令デコードステージ3は、命令デコーダおよびパイプラインシーケンサを含む。命令デコーダは命令フェッチステージ2によりフェッチされた複数の命令を受けて解読する。パイプラインシーケンサ（命令スケジューラ）はこのデコードされた複数の命令のたとえばマシンタイプを識別し、異なるマシンタイプの命令を同時に対応の機能ユニットへ投入する。ここで、マシンタイプとは、命令がどの機能ユニットで処理されるべきかを示す情報である。

機能ユニット4ないし7は、それぞれパイプライン化されており、与えられた命令をクロック信号に応答して実行する。この第31図においては、4つの機能ユニットが一例として示されており、最大4つの命令が並列処理可能である。機能ユニット4および5は整数加算などを行なうための整数演算ユニットであり、整数演算を行なう実行ステージ（EX）および書込ステージ（WB）を含む。書込ステージ（WB）は、処理の実行結果をデータレジスタ（図示せず）へ書込む。

機能ユニット6は、データメモリ8へのアクセス（データのロードまたはストア）を実行するユニットであり、アドレス生成ステージ（ADR）、データメモリ8へのアクセス実行ステージ（ME

M）およびデータレジスタ（図示せず）へのデータの書込ステージ（ＷＢ）である。ここで書込ステージ（ＷＢ）においてはデータのロードまたはデータのストアが行なわれる。

機能ユニット７は、浮動小数点加算などを実行するためのユニットであり、３段の実行ステージ（ＥＸ１、ＥＸ２およびＥＸ３）と、実行結果のデータレジスタへの書込を行なう書込ステージ（ＷＢ）とを含む。浮動小数点数は指数と仮数とを用いて表わされ、その小数点の位置が一定しない数であり、浮動小数点演算は整数演算などに比べてその実行により多くのサイクルが必要とされる。

この並列処理装置においては、各段はすべてパイプライン化されており、命令フェッチ、命令デコードおよび命令実行ならびにデータ書込が互いにオーバラップしつつ実行される。したがって、命令メモリからフェッチされた命令が次のサイクルで命令デコードステージでデコードされる。次に動作について簡単に説明する。

命令デコードステージ３は、命令フェッチステージ２へ命令フェッチ要求を出力する。命令フェッチステージ２は、この命令フェッチ要求に応答して命令ポインタＩＦ＿ＰＣを命令メモリ１へ与え、命令メモリ１からその命令ポインタＩＦ＿ＰＣに対応する複数の命令をフェッチする。このフェッチされた複数の命令は命令デコードステージ３へ同時に与えられる。命令デコードステージ３はこの与えられた複数の命令を同時にデコードする。この命令デコードステージ３はさらに、デコードされた命令から、計算リソースおよびデータレジスタが競合しない、並列処理可能な命令を検出し、この並列処理可能な命令を対応の機能ユニットへ発行（投入）する。命令を与えられた機能ユニットはこの与えられた命令に従って処理を並列に実行する。機能ユニット４ないし７の処理の実行は、パイプライン化されており、この第３１図に示す各ステージに従って処理が実行される。また、命令フェッチステージ２および命令デコードステージおよび命令実行ステージ（機能ユニット４ないし７）の動作はパイプライン化されており、それぞれ互いにオーバラップしながら所定の動作を実行している。

上述のように各ステージの動作をパイプライン化し、かつ複数の機能ユニットで処理を並列に実行することにより高速で命令を実行することができる。

［発明が解決しようとする課題］

上述のように、スーパースカラーにおいては、複数の命令がフェッチされ、かつ同時に複数の命令が実行されるため、通常のコンピュータに比べて処理速度の向上を図ることができる。たとえば、第３１図に示す構成において、同時にフェッチされた４つの命令が４つの機能ユニット４ないし７で並列に実行された場合、４クロックサイクルで４命令を処理することができる（機能ユニット４、５および６のパイプラインが機能ユニット７の処理完了まで待ち合わせ状態となる場合）。

命令スケジューラ（または命令デコードステージに含まれるパイプラインシーケンサ）は、効率的に並列処理が実行されるように命令のスケジューリングを実行するが、同時にフェッチされた命令が常に同時に処理されるとは限らない。たとえば、以下のようなデータ依存関係がある命令を考える。

① ａｄｄ　Ｒ１，Ｒ２，Ｒ３
；Ｒ２＋Ｒ３＝Ｒ１
② ｓｕｂ　Ｒ４，Ｒ１，Ｒ５
；Ｒ１－Ｒ５＝Ｒ４）

命令①はレジスタＲ２の内容にレジスタＲ３の内容を加え、その加算結果をレジスタＲ１に書込む命令である。

命令②はレジスタＲ１の内容からレジスタＲ５の内容を減算し、この減算結果をレジスタＲ４に書込む命令である。このような命令①および②の演算はたとえば「ｘ＋ｙ－ｚ」という処理に対応する。

この命令①および②が同時にフェッチされたとき、これらはレジスタＲ１を共通に使用しており、命令②は命令①の結果を使用することになるため、

これらの命令①および②は同時に実行することができない。このような命令間のデータ依存関係が存在する場合、命令の発行状況（命令デコードステージから機能ユニットへの命令の発行態様）は第３２図に示すようになる。

第３２図においては、同時に処理可能な命令のみが発行される状況が示されており、また同時に発行される命令は、アドレスの小さい方（図の左側に位置する）の命令から順に決定される。

第３２図において、サイクル１においては、命令２．３および４が相互にまたは命令１に対しデータ依存関係を有するため、これらは発行することができず、命令１のみが発行される。

サイクル２においては、命令４は命令２および／または命令３とデータ依存関係があり、命令２および命令３は互いにデータ依存関係を有しないため、命令２および３が発行される。

サイクル３においては、残った命令４が発行される。

サイクル４において、新たにフェッチされた４つの命令５ないし８のデコードが行なわれ、相互にデータ依存関係のない命令５および６が発行される。

サイクル５において命令７および８が相互にデータ依存関係を有しないためこれらが発行される。ここで、第３２図においては□印で囲んだ数字は、相互にデータ依存関係のない発行可能な命令を示している。

この第３２図に示すような、同時にフェッチされた複数の命令がすべて発行されるまで次の命令のフェッチを待ち合わせる方式の場合、最初にフェッチされた命令１ないし４と、次にフェッチされた命令５ないし８がすべて発行されるまでに５サイクルが必要とされる。したがって、このような命令供給および発行方式では、パイプラインに空きが生じ、並列処理装置の高速処理性能が損なわれる。

このような場合もし、第３２図において命令４と命令５および６との間にデータ依存関係が存在せず、かつ任意の数の命令を命令メモリからフェッチすることが可能であれば、第３３図に示す命令発行方式を実現することができる。第３３図は改良された命令発行方式における命令発行状況を示す図である。

第３３図を参照して、サイクル１において、同時にフェッチされた４つの命令１ないし４のうち、命令１が発行される。

サイクル２において、新たに命令５が供給され、命令２ないし５のデコードが行なわれる。この４つの命令２ないし５のうち、依存関係のない命令２および３が発行される。

サイクル３において、新たに命令６および７が供給され、命令４ないし７がデコードされる。このデコード結果に従って命令４、５および６が発行される。

サイクル４において、新たに命令８，９および１０が供給され、命令７ないし１０のデコードが行なわれ、このデコード結果に従って命令７および８が発行される。

この第３３図に示す命令発行方式においては、命令１ないし８の８個の命令の発行に４サイクルが必要とされるだけであり、第３２図に示す方式に比べてより高速で命令の処理実行を行なうことができる。この第３３図に示す命令発行手順を実現する手法の１つとして第３４図に示す方法が考えられる。第３４図は第３３図に示す命令発行状況を実現するための命令の供給手順を示す図である。

第３４図を参照して、ステップ（１）において、命令レジスタ（命令を保持するレジスタ）に保持された命令２ないし５のうち命令２および３が発行されると命令レジスタに空きが生じる。

ステップ（２）において、この空きの生じたレジスタの数だけ命令レジスタの内容をシフトさせる。すなわち第３４図のステップ（２）において命令４および５は左へ２つだけレジスタ位置がシフトされる。

ステップ（３）において、この空きの生じた命令レジスタ内へ次の命令６および７がフェッチされる。この第３４図に示すステップ（１）ないし

（3）を1サイクルで実行する必要がある。この第34図に示す命令シフト動作を実行する構成としては第35図に示すような構成を考えることができる。

第35図において、命令デコードステージは、命令を格納する命令レジスタIR1～IR8と、与えられた命令をシフトさせるバレルシフタBRと、命令デコーダIDとを含む。命令レジスタIR1～IR4は命令フェッチステージ2によりフェッチされた命令およびバレルシフタからシフトされた命令を格納する。命令レジスタIR5ないしIR8は、命令デコーダIDへ与えられた命令がデコードされずに格納される。バレルシフタBRは、命令レジスタIR5ないしIR8からの命令のうち、命令デコーダIDからの発行命令数情報に従って命令をシフトさせる。

通常、並列演算処理装置は第36図に示すように2相の互いに重なり合わないクロック信号TおよびLに従って動作している。次に動作について簡単に説明する。

前のサイクルでフェッチされた命令がクロック信号Tに応答して命令レジスタIR1～IR4で保持される。命令レジスタIR1～IR4の保持命令は命令デコーダIDへ与えられ、デコードされ、該デコード結果に従って機能ユニットへ発行される。

一方、命令レジスタIR5～IR8はクロック信号Lに応答してその命令デコードIDへ与えられた命令を保持しバレルシフタBRへ与える。バレルシフタBRは、命令デコーダIDからの発行命令数情報に応答して、この命令レジスタIR5～IR8から与えられた命令をシフトさせる。このバレルシフタBRの内容は、次のクロック信号Tに応答して命令レジスタIR1～IR4へ与えられて保持される。

この場合、バレルシフタはクロック信号Lから次のクロック信号Tの間にその命令のシフト動作を完了していなければならない。しかしながら、バレルシフタBRにおいては、命令長が長く（たとえば32ビット）、高速でその命令シフト動作を実行することができず、命令シフト動作に長時間を要する。したがって、次のクロック信号Tに応答して命令レジスタIR1～IR4へ命令を与えて保持させることができなくなるため、高速で命令デコーダIDへ命令を供給することができなくなり、並列処理装置の高速動作性が損なわれる。

また、命令メモリは、この命令レジスタIR1～IR4においていくら空きレジスタが存在しているかという情報を知り、この情報に基づいて命令供給数および命令供給位置を定める必要がある。この命令レジスタIR1～IR4においていくつ空きレジスタが存在するかの情報は命令デコーダからの発行命令数情報で得られるが、この命令発行数情報が発生されるのは命令デコーダIDにおけるデコード動作後であり、また、この情報に従って、命令メモリにおいて命令供給数および命令供給位置を判定するのに長時間を要し、またその判定動作開始タイミングも遅い。このため、所望の命令をフェッチするのに長時間を要することになり、パイプライン構成を乱さずに実行するためには、クロック信号TおよびLの周期が長くなり、並列演算処理装置の高速動作性が損なわれる。

それゆえ、この発明の目的は並列処理を効率的に行なうことのできる並列処理装置を提供することである。

この発明の他の目的は並列処理を効率的に行なうことのできる命令供給装置を備えた並列処理装置を提供することである。

[課題を解決するための手段]

この発明に係る並列処理装置における命令供給装置は、命令メモリ装置からの命令の供給の有無を示す信号と、リセット信号と、分岐が発生したことを示す分岐発生指示情報とに応答して、命令の有効／無効を示す有効性フラグを形成する手段と、命令メモリから同時にフェッチされた複数の命令と、この同時にフェッチされた複数の命令の上記命令メモリにおけるアドレスと有効性フラグとを同時に格納する記憶手段を含む。この記憶手段は、1エントリが1命令と、対応のアドレスと、対応の有効性フラグとからなる複数のエントリ格

納領域を備える。またこの記憶手段は、各々に異なる書込エントリ内容が同時に伝達される複数の入力ポートと、各々に異なる読出エントリ内容が同時に伝達される複数の出力ポートと、書込キューアドレスおよび読出キューアドレスに応答して複数の書込エントリおよび複数の読出エントリを複数の入力ポートおよび複数の出力ポートへそれぞれ接続する手段を含む。

この命令供給装置はさらに命令デコーダからの命令発行数情報と分岐発生指示情報と命令供給有無指示信号とに応答して、上記書込キューアドレスおよび読出キューアドレスをそれぞれ生成する手段を備える。

この記憶手段から同時に読出された複数のエントリの内容が命令デコード装置へ与えられる。

〔作用〕

記憶装置は、互いに独立にアクセス可能な複数の入力ポートおよび複数の出力ポートを含む。この複数の入力ポートへは同時に命令メモリからフェッチされた複数の命令、アドレスおよび有効性フラグを伴なってそれぞれの入力ポートを介して異なるエントリ領域へ同時に格納される。

またこの複数のエントリ格納領域からは読出キューアドレスに応答して同時に複数のエントリ領域が選択されそれぞれ異なる出力ポートに接続され、これにより複数のエントリ内容が同時に読出される。

この入力ポートおよび出力ポートはそれぞれ独立にアクセス可能であるため、エントリへの書込読出を同時に実行することができる。また書込キューアドレスおよび読出キューアドレスは、それぞれ命令の発行状況および命令メモリからの命令供給状況に応じて選択され、常に同一数の命令が発行対象命令として命令デコード装置へ与えられる。有効性フラグは該命令の発行の可否を決定する。

これにより、命令デコード装置へは、常に発行された命令数を補償する命令が同時に与えられ、効率的に命令供給を実行することができることになり、命令実行を高速で行なうことが可能となる並列処理装置を実現する。

〔発明の実施例〕

第１図はこの発明の一実施例であるスーパースカラーの要部の構成を示す図である。第１図に示す構成はスーパースカラーの命令デコード・ステージに対応する。この第１図に示す装置は、２相の互いに重なり合わないクロック信号ＴおよびＬ（第３６図参照）に応答して動作する。

図示しないが命令フェッチ・ステージ（第３６図参照）は、この命令デコード・ステージからの命令フェッチ要求に従って複数（本実施例では４つ）の命令を命令メモリからフェッチする。フェッチした命令ＩＣ＿ｄａｔａおよびこのフェッチした命令の論理アドレスＩＦ＿ＰＣ＿Ｌは、命令が命令メモリから供給されることを示す信号ＩＣ＿ｒｅａｄｙとともにこのデコード・ステージへ供給される。フェッチ命令の論理アドレスＩＦ＿ＰＣ＿Ｌは、命令フェッチ・ステージに含まれる、命令メモリのアドレスを作成するポインタから供給される。

命令メモリは、たとえばキャッシュミス等により低速の主メモリ（または外部メモリ）へアクセスしている状態または命令メモリがアクセスを受けている状態等においては、命令メモリが命令フェッチ要求を受入れられないことを示す信号ＩＣ＿ｂｕｓｙを発生してデコード・ステージへ与える。

第１図を参照して、この発明の一実施例である命令供給装置は、命令メモリからフェッチされた命令の論理アドレス（ＩＦ＿ＰＣ＿Ｌ）、フェッチされた命令（ＩＣ＿ｄａｔａ）およびこの命令の有効／無効を示す有効性フラグ（ｉｎｓｔ＿ａｖａｉｌ）を１エントリとして格納するランダム・アクセス・メモリ（ＲＡＭ）からなるキュー９を含む。このキュー９は、後にその構造を詳細に説明するが、たとえば１２エントリを格納することができ、かつ複数の書込ポートおよび複数の読出ポートを有する。この複数の書込ポートおよび複数の読出ポートにより、１つの書込ゲートおよび読出ゲートを指定することにより、連続する複

数（本実施例では４つ）のエントリの内容を同時に書込または読出すことができる。

このキュー９への命令の書込および読出ならびに格納された命令の有効無効を制御するために、クロック信号Ｔに応答して、与えられた信号を保持するＴラッチ回路１０，１１，１２，１３および２８と、各Ｔラッチ回路１０～１３および２８に対応して設けられ、対応のＴラッチ回路の出力データ（保持データ）をクロック信号Ｌに応答して保持するＬラッチ回路１７，１８，１９，２０および２９と、各Ｌラッチ回路１７～２０および２９の保持データと命令デコード２６からの信号ｂｒ＿ｔａｋｅｎ、ｉｓｓｕｅｄ＿ｉｎｓｔ＿ｃｏｕｎｔ、およびｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙとから、キュー９の読出アドレス、書込アドレス、動作制御信号および命令の有効／無効を示す信号を発生する論理回路３０，３１，３２、３３、３４、３５および３６ならびに５０を含む。

Ｔラッチ回路１０（以下、キュートップラッチ１０と称す）は、キュー９に命令メモリからフェッチされた命令を書込むべき複数のエントリの最初のアドレスを示すアドレスｑｕｅｕｅ＿ｔｏｐをクロック信号Ｔに応答して保持する。すなわち、キュー９に４つの命令が格納される場合、命令、その論理アドレスおよび有効性フラグはキュー９のアドレス：

ｑｕｅｕｅ＿ｔｏｐ，ｑｕｅｕｅ＿ｔｏｐ＋１
ｑｕｅｕｅ＿ｔｏｐ＋２，ｑｕｅｕｅ＿ｔｏｐ＋３

に書込まれる。ここで「＋」はキュー９が有する全エントリ数（本実施例では１２）についてのモジュロ加算である。

Ｌラッチ回路１７（以下、キュートップＬラッチ１７と称す）は、キュートップラッチ１０の保持データを、クロック信号Ｌに応答して保持する。

Ｔラッチ回路１１（以下、スコープラッチ１１と称す）は、クロック信号Ｔに応答して、キュー９から同時に読出されるべき４エントリの最初の１エントリが登録されているキュー９のアドレスをクロック信号Ｔで保持する。すなわち、キュー９のアドレス：

ｓｃｏｐｅ，ｓｃｏｐｅ＋１，
ｓｃｏｐｅ＋２，ｓｃｏｐｅ＋３

のエントリの内容が読出される。ここで、「＋」はキュー９の全エントリ数（本実施例では１２）についてのモジュロ加算である。

Ｔラッチ回路（以下、命令フェッチラッチと称す）１２は、命令フェッチを行なうことを示すフラグＩＣ＿ｆｅｔｃｈをクロック信号Ｔに応答して保持する。この命令フェッチ要求フラグＩＣ＿ｆｅｔｃｈが"１"のサイクルにおいては、命令フェッチ要求が命令フェッチステージへ発行される。

Ｌラッチ回路（以下、命令フェッチＬラッチと称す）１９は、クロック信号Ｌに応答して、命令フェッチラッチ１２にラッチされたフラグＩＣ＿ｆｅｔｃｈを保持する。

Ｔラッチ回路（以下、命令ミスラッチと称す）１３は、命令メモリからの命令の待機状態であることを示すフラグＩＣ＿ｍｉｓｓをクロック信号Ｔに応答して保持する。

Ｌラッチ回路（以下、命令ミスＬラッチと称す）２０は、この命令ミスラッチ１３の保持するフラグＩＣ＿ｍｉｓｓをクロック信号Ｌに応答して保持する。

Ｔラッチ回路（以下、初期状態ラッチと称す）２８は、クロック信号Ｔに応答してキュー９の初期状態を表わすフラグｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅを保持する。

Ｌラッチ回路（以下、キュー初期状態Ｌラッチと称す）２９は、キュー初期状態ラッチ２８の保持するフラグｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅをクロック信号Ｌに応答して保持する。

論理回路３０（以下、キュートップロジック３０と称す）は、命令デコーダ２６からの分岐発生信号ｂｒ＿ｔａｋｅｎと、キュー初期状態Ｌラッチ２９の保持するキュー初期状態フラグｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅと、信号ＩＣ＿ｒｅａｄｙと、キュートップＬラッチ１７からのキュー９の書込先頭アドレスｑｕｅｕｅ＿ｔｏｐと、命

令フェッチLラッチ１９からのフラグＩＣ＿ｆｅｔｃｈと、論理回路３１からの次のサイクルにおけるキュー９の読出先頭アドレスｓｃｏｐｅ＿ｎｅｘｔと、キュー９が利用可能（書込可能）であるか否かを示す論理回路３４からのフラグｑｕｅｕｅ＿ａｖａｉｌおよび命令デコーダ２６から発生される、分岐命令が登録されたエントリのアドレスｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙとに応答して、次のサイクルにおけるキュー９の書込先頭アドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎを形成し、キュートップラッチ１０、論理回路３４および５０へ与える。

論理回路３１（以下、スコープネクストロジック３１と称す）は信号ｂｒ＿ｔａｋｅｎと、スコープＬラッチ１８からの読出先頭アドレスｓｃｏｐｅ＿Ｌと、命令デコーダ２６からの発行命令数データｉｓｓｕｅｄ＿ｉｎｓｔ＿ｃｏｕｎｔおよびアドレスｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙとに応答して、次のサイクルにおけるキュー９の書込先頭アドレスｓｃｏｐｅ＿ｎｅｘｔを発生するとともに、このアドレスｓｃｏｐｅ＿ｎｅｘｔが１１より大きいことを示すフラグｓｃｏｐｅ＞１１を発生する。次のサイクルにおけるアドレスｓｃｏｐｅを示す信号ｓｃｏｐｅ＿ｎｅｘｔはスコープラッチ１１および論理回路５０へ与えられるとともに論理回路３４へも与えられる。フラグｓｃｏｐｅ＞１１は論理回路５０へ与えられる。

論理回路３３（以下、命令ミスロジック３３と称す）は、命令フェッチLラッチ１９からのフラグＩＣ＿ｆｅｔｃｈ＿Ｌと、命令ミスLラッチ２０からのフラグＩＣ＿ｍｉｓｓ＿Ｌと、フラグｂｒ＿ｔａｋｅｎと、信号ＩＣ＿ｒｅａｄｙとに応答して、、命令待ち状態であるか否かを示す信号ＩＣ＿ｍｉｓｓを作成して命令ミスラッチ１３へ与える。

論理回路３６（以下、キュー初期状態ロジック３６と称す）は、キュー初期状態Ｌラッチ２９からの信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＿Ｌと、信号ｂｒ＿ｔａｋｅｎと、命令ミスLラッチ２０からの信号ＩＣ＿ｍｉｓｓ＿Ｌと、命令フェッチLラッチ１９からの信号ＩＣ＿ｆｅｔｃｈ＿Ｌと、信号ＩＣ＿ｒｅａｄｙとに応答してキュー９が初期状態であるかを示す信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅを作成してキュー初期状態ラッチ２８へ与える。

論理回路３４（以下、キューアベールロジック３４と称す）は、キュートップＬラッチ１７からのアドレスｑｕｅｕｅ＿ｔｏｐ＿Ｌと、スコープネクストロジック３１からのアドレスｓｃｏｐｅ＿ｎｅｘｔと、信号ｂｒ＿ｔａｋｅｎと、キュートップロジック３０からのアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎ、論理回路５０からのキュー９の状態（第１、第２および第３の状態：これは後に説明する）を示す信号ｑｕｅｕｅ＿ｓｔａｔｅとに応答して、キュー９が利用可能であるか否かを示す信号ｑｕｅｕｅ＿ａｖａｉｌを発生し、キュートップロジック３０へ与える。

論理回路３２（以下、命令フェッチロジック３２と称す）は、命令メモリまたは命令フェッチステージから与えられる、命令メモリが命令フェッチ要求を受入れられない状態を示す信号ＩＣ＿ｂｕｓｙに応答して、命令メモリへフェッチ要求を出すか否かを示す信号ＩＣ＿ｆｅｔｃｈを発生し、命令フェッチラッチ１２へ与える。

論理回路３５（以下、命令アベールロジック３５と称す）は、信号ＩＣ＿ｒｅａｄｙと、信号ＩＣ＿ｆｅｔｃｈ＿Ｌと、信号ｂｒ＿ｔａｋｅｎと、信号ＩＣ＿ｍｉｓｓ＿Ｌとに応答して、キュー９に含まれる命令が有効であるか否かを示すフラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎを発生し、キュー９のエントリ内の有効性フラグｉｎｓｔ＿ａｖａｉｌの値を決定する。

論理回路５０（以下、キューステートロジック５０と称す）は、命令デコーダ２６からのアドレスｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙと、キュートップロジック３０からのフラグｑｕｅｕｅ＿ｔｏｐ＞１１と、スコープネクストロジック３１からのフラグｓｃｏｐｅ＞１１とアドレスｓｃｏｐｅ＿ｎｅｘｔとに応答して、キュー９が、第１の状態、第２の状態および第３の状態のいずれに

あるか示す信号ｑｕｅｕｅ＿ｓｔａｔｅを発生し、キューアベールロジック３４へ与える。ここで第１の状態はキュー９が初期状態にあり、その書込先頭アドレスｑｕｅｕｅ＿ｔｏｐおよび読出先頭アドレスｓｃｏｐｅがともに０または同一であり、またアドレスｑｕｅｕｅ＿ｔｏｐからの４エントリの有効性フラグｉｎｓｔ＿ａｖａｉｌがすべてオフ（０）の状態を示す。この状態はリセット状態、分岐発生状態および命令到着待ち状態のいずれかの状態である。第２の状態は、読出先頭アドレスｓｃｏｐｅが書込先頭アドレスｑｕｅｕｅ＿ｔｏｐ以下の状態、すなわち、

ｓｃｏｐｅ≦ｑｕｅｕｅ＿ｔｏｐ

の状態でありノーマル（ＮＯＲＭＡＬ）状態である。

第３の状態は書込先頭アドレスｑｕｅｕｅ＿ｔｏｐが読出先頭アドレスｓｃｏｐｅ以下の場合であり、すなわち、

ｑｕｅｕｅ＿ｔｏｐ≦ｓｃｏｐｅ

の状態を示す。この第３の状態はリバース（ＲＥＶＥＲＳＥ）状態と呼ばれる。この第２の状態および第３の状態を区別することにより、アドレスｓｃｏｐｅ＿ｎｅｘｔとアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎが等しくなった状態において、キュー９が空の状態であるのか満杯（フル）状態であるのかを区別することができる。すなわち、第２の状態からｓｃｏｐｅ＿ｎｅｘｔ＝ｑｕｅｕｅ＿ｔｏｐ＿ｉｎとなった場合、このときはキュー９は空の状態を示している。また、第３の状態からｓｃｏｐｅ＿ｎｅｘｔ＝ｑｕｅｕｅ＿ｔｏｐ＿ｉｎとなった場合、このときはキュー９はフル状態となる。この第２および第３の状態については後に詳細に説明する。

キュー９と命令デコーダ２６との間には、キュー９から読出された複数のエントリの内容をクロック信号Ｌに応答して保持するＬラッチ回路１４、１５および１６が設けられる。Ｌラッチ回路１４（以下、ＩＲＬラッチ１４と称す）は、キュー９から同時に読出された複数の命令ｉｎｓｔを保持する。

Ｌラッチ回路１５（以下、ＰＣＬラッチ１５と称す）は、キュー９から同時に入力された命令に対応するアドレスＰＣを保持する。

Ｌラッチ回路１６（以下、命令アベールＬラッチ１６と称す）は、キュー９から読出された命令に付随する有効性フラグｉｎｓｔ＿ａｖａｉｌを保持する。

命令デコーダ２６は、このＩＲＬラッチ１４、ＰＣＬラッチ１５および命令アベールＬラッチ１６からのアドレス、命令および有効性フラグに応答して、与えられた複数の命令をデコードし、並列実行可能な命令を検出し、その命令を機能ユニットへ発行する。この同時に機能ユニットへ発行される命令は、ＰＣＬラッチ１５からのアドレスの小さい順に決定され、複数の命令が依存関係を有する場合には、アドレスの小さい方の命令が先に発行される。

次に、この発明による命令フェッチ方式の基本的な動きについて説明する。

第２図は、キュー９の概念的構成を示す図である。第２図において、キュー９は、アドレス０ないし１１の付された記憶領域を有し、各アドレスに対応して１個のエントリＥが格納される。各エントリＥは、命令メモリからフェッチされた命令のアドレス（論理アドレス）を格納する領域Ⅰと、命令メモリからフェッチされた命令ＩＣ＿ｄａｔａを格納する領域Ⅱと、この対応の命令が有効であるか否かを示すフラグｉｎｓｔ＿ａｖａｉｌを格納する領域Ⅲとを含む。このアドレスＩＦ＿ＰＣ＿Ｌ、命令ＩＣ＿ｄａｔａおよびフラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎはそれぞれ４つの命令が並列にこのキュー９に伝達され、４つの異なるアドレス領域に同時に格納される。このキュー９のエントリの登録領域はアドレスｑｕｅｕｅ＿ｔｏｐにより指定され、アドレスｑｕｅｕｅ＿ｔｏｐ～ｑｕｅｕｅ＿ｔｏｐ＋３の領域に格納される。

このキュー９からの命令の読出は、４つの命令が並列に読出されることにより行なわれる。この読出のアドレスの指定は、アドレスｓｃｏｐｅにより行なわれ、アドレスｓｃｏｐｅ～ｓｃｏｐｅ

＋３のアドレスに登録されたエントリが並列に同時に読出される。次に、第３図を参照してこの発明による命令フェッチ方式の基本的な動作について説明する。

まず状態（１）はリセット直後または初期化直後の初期状態（第１の状態）である。この初期状態においては、アドレスｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅはともにキュー９の初期アドレス０を示している。このとき、キュー９においてはｑｕｅｕｅ＿ｔｏｐすなわちアドレス０から始まる有効性フラグはオフ（０）とされる。この状態においては、有効性フラグがオフ（０）を示しているためそのキュー９のアドレス０～３の４エントリは無効であり、発行または実行が禁止されていることを示している。

状態（２）において、アドレスｑｕｅｕｅ＿ｔｏｐが示すアドレス０から始まる４つのアドレスに対して命令が書込まれる。この書込まれた命令は有効な命令であるため、これに対応する有効性フラグがオン（１）とされ、この書込まれた命令が有効であることが示され、この４つのエントリの命令の読出、発行／実行可能なことが示される。この状態（２）においては、まだｑｕｅｕｅ＿ｔｏｐの変更は行なわれていない。この状態（２）においては、アドレスｓｃｏｐｅが示すアドレス０から始まる４つのエントリが読出され、命令デコーダへ与えられ、解読される。

状態（２）において、命令デコーダから２つの命令が同時処理可能であるとして機能ユニットへ発行され、読出先頭アドレスｓｃｏｐｅがアドレス２へ変更される。

状態（３）のサイクルにおいては、書込先頭アドレスｑｕｅｕｅ＿ｔｏｐがアドレス４に変更されており、このアドレス４から始まる４つのアドレス領域に次の４つの命令４ないし７がそれぞれ格納される。この新たに書込まれた命令４ないし７に対応する有効性フラグも同時にオンとされる。

状態（４）は、１つの命令が発行された後の状態を示している。１命令が発行されたため、読出先頭アドレスｓｃｏｐｅは１つ移動し、アドレス３を示す。またこの状態（４）においては、新たに４つの命令８ないし１１が書込先頭アドレスｑｕｅｕｅ＿ｔｏｐが示すアドレス８から連続する４つのアドレス領域にそれぞれ書込まれており、対応の有効性フラグもオン（１）とされる。この状態（４）のサイクルにおいては、アドレスｓｃｏｐｅが示すアドレス３から始まる４つのアドレスの命令がキュー９から読出され命令デコーダへ与えられて解読される。

状態（５）は、前のサイクルにおいて命令デコーダからは１つも命令が発行されなかった場合の状態で示している。この場合、発行された命令数は０であるため、書込先頭アドレスｓｃｏｐｅは移動せず、アドレス３を示す。この状態（５）のサイクルにおいては、アドレスｓｃｏｐｅが示すアドレス３から始まる４つのアドレスの命令が再びキュー９から読出されて命令デコーダにより解読される。この状態（５）においては、先のサイクルにおいてｑｕｅｕｅ＿ｔｏｐ＋４（１２のモジュロ加算）が行なわれ、書込先頭アドレスｑｕｅｕｅ＿ｔｏｐは０（８＋４＝０）となり、アドレス０を示している。この場合、ｑｕｅｕｅ＿ｔｏｐから始まる４つのアドレス領域は、まだ発行されていない命令（アドレス３の命令）を含んでいるため、キュー９への命令の取込は行なわれない。

状態（６）は、前のサイクルで命令デコーダから３つの命令が発行された場合の状態を示している。３命令が発行されたため、読出先頭アドレスｓｃｏｐｅは３シフトされ、アドレス６を示している。書込先頭アドレスｑｕｅｕｅ＿ｔｏｐが示すアドレス０から始まる４つのアドレス領域に命令１２，１３，１４および１５が格納される。この命令１２ないし１５は有効であるため、それに対応する有効性フラグがオン状態とされる。このキュー９において空き領域があるか否かの判定は、キューアベールロジック３４から出力される信号ｑｕｅｕｅ＿ａｖａｉｌにより行なわれる。

次に、分岐命令による分岐が生じた場合の基本的動作についてそのフローを示す第４図を参照し

て説明する。いまキュー９が状態（１１）にあった場合を想定する。この状態（１１）においては、アドレスｑｕｅｕｅ＿ｔｏｐはアドレス１０を示しており、アドレスｓｃｏｐｅはアドレス２を示している。この状態（１１）においては、アドレス１０，１１，０，１に命令８，９，１０および１１が書込まれる。キュー９のアドレス２，３，４および５に格納されている命令２，３，４，５が読出されて命令デコーダへ与えられる。この状態（１１）のサイクルにおいて、アドレス４の命令２が分岐命令であり、この分岐命令２に従って分岐が生じたとする。次の状態（１２）においては以下のことが行なわれる。

キュー９のアドレス４以降の内容は、命令２において分岐が生じたためもはや不要である。

アドレスｑｕｅｕｅ＿ｔｏｐに続くアドレスすなわちアドレス４ないし７に格納された４エントリの有効性フラグをオフ（０）に設定する。アドレスｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅをキュー９のアドレス４に設定する。

また、分岐先命令のアドレスは、このサイクルで命令メモリへ送られ、分岐先命令はその次のサイクルで命令メモリから供給される。すなわち、分岐先命令はこのサイクルではキュー９には書込まれない。

アドレスｓｃｏｐｅより始まる４エントリの命令は発行対象として解読することができる。しかしながら、これらの内容は機能ユニットへ発行してはならないものであるため、このサイクルにおいては、アドレスｑｕｅｕｅ＿ｔｏｐより始まる４エントリの有効性フラグがオフ（０）に設定される。

状態（１３）においては、アドレスｑｕｅｕｅ＿ｔｏｐから始まるアドレス４ないし７の領域に分岐先命令２０～２３が命令メモリから供給されて書込まれる。またこの命令２０ないし２３は分岐先命令であり、解読・発行されるべきものであるため、その有効性フラグもオン（１）に設定される。

このサイクルにおいては、またアドレスｓｃｏｐｅ（キューアドレス４）から始まる４エントリの命令２０ないし２３が命令デコーダへ与えられ、解読される。

次に、プロセサからの命令フェッチ要求に対して、命令メモリから命令を供給することができない場合の基本的な動きについてそのフローを示す第５図を参照して説明する。

いま、キュー９が第５図に示す状態（２１）にあった場合を想定する。この状態（２１）においては、キュー９のアドレス０ないし３へ命令８ないし１１が書込まれ、一方、キュー９のアドレス６，７，８および９に格納されている命令２，３，４および５が命令デコーダへ与えられ、発行対象として解読される。

このサイクルにおいて命令２，３および４が機能ユニットへ発行されたとする。

このサイクルにおいて、命令メモリに対して行なった命令フェッチ要求に対して、命令メモリは何らかの原因（たとえば命令メモリがキャッシュで構成される場合、キャッシュミス）によりこのサイクル内で命令を供給できなかったとする。

状態（２２）においては、前のサイクルで命令２，３および４が発行されているため、アドレスｓｃｏｐｅが３つシフトしてアドレス９を示している。

また、アドレスｑｕｅｕｅ＿ｔｏｐはアドレス４を示している。命令メモリから命令が供給されていないため、このキュー９のアドレス４，５，６および７に格納されている命令はフェッチ要求が出された命令と異なるため、解読してはならない。そこで、命令が命令メモリから供給されない場合、このアドレスｑｕｅｕｅ＿ｔｏｐに続くアドレスに格納された４エントリの有効性フラグがオフ（０）に設定される。

状態（２３）においてもまだ命令メモリから命令が供給されず、プロセサは命令到着待機状態にある。このためキュー９への命令書込は停止する。一方、命令フェッチが行なわれなくても、キュー９に未発行の命令が存在すれば命令デコーダを介して機能ユニットへの命令発行が続行される。こ

の第５図に示す例においては、前のサイクル（状態（２２））において２命令が発行されており、このアドレスｓｃｏｐｅは１１を示している。

状態（２４）において、命令が命令メモリから到着したため、アドレスｑｕｅｕｅ＿ｔｏｐ（アドレス４）から始まるキュー９の４エントリに命令が書込まれる。すなわち、命令１２～１５がアドレス４～７にそれぞれ書込まれる。

また前のサイクルにおいて２つの命令（命令７および８）が発行されており、アドレスｓｃｏｐｅはアドレス１を示す。

次に、第３図ないし第５図に示す動作を実現するためのロジック３０ないし３６の論理について説明する。まず、キュートップロジック３０が実現するロジックについて説明する。

（１）　初期状態：この状態は次の２つの状態を含む。

（ａ）　リセット状態：信号ｒｅｓｅｔ＝１

このとき、ｑｕｅｕｅ＿ｔｏｐ＝０となる。

（ｂ）　初期化状態：信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＿Ｌ＝１

この状態は、キュー９の書込先頭アドレスはキュートップＬラッチ１７の保持するアドレスに設定される。したがって、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿Ｌ

となる。

次に非初期状態、すなわちｒｅｓｅｔ・ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＿Ｌ＝０の場合について説明する。ここで「・」は論理積演算を示す。

（２）　分岐発生状態：信号ｂｒ＿ｔａｋｅｎ＝１：

この状態においては、アドレスｑｕｅｕｅ＿ｔｏｐは、キュー９において分岐命令が格納されたアドレスに設定される。したがって、このときは、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙ

となる。

（３）　命令フェッチ要求が出されても命令供給が行なわれない場合：ＩＣ＿ｆｅｔｃｈ＿Ｌ＝１かつＩＣ＿ｒｅａｄｙ＝０

このとき、アドレスｑｕｅｕｅ＿ｔｏｐは前のサイクルと同じ値を維持するので、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿Ｌ

となる。

（４）　発行された命令フェッチ要求に対し命令供給が行なわれ、かつキュー９に空きが存在する場合

：ＩＣ＿ｆｅｔｃｈ＿Ｌ＝１かつＩＣ＿ｒｅａｄｙ＝１、かつｑｕｅｕｅ＿ａｖａｉｌ＝１

このとき、アドレスｑｕｅｕｅ＿ｔｏｐは次のサイクルにおいて４シフトされるため、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＋４

となる。ここで、「＋」はモジュロ１２の加算である。したがって、

ｉｆ（ｑｕｅｕｅ＿ｔｏｐ＞１１）

｛ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ－１２｝

となる。

（５）　命令フェッチ要求が出され、命令供給が行なわれても、キューに空き領域（４エントリ分）が存在しない場合

：ＩＣ＿ｆｅｔｃｈ＿Ｌ＝１かつＩＣ＿ｒｅａｄｙ＝１かつｑｕｅｕｅ＿ａｖａｉｌ＝０

この状態はキュー９に４エントリ分の命令を書込む領域が存在しないため、命令の書込が行なわれない状態を示しており、アドレスｑｕｅｕｅ＿ｔｏｐは前のサイクルと同じ状態を維持する。したがって、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿Ｌ

となる。

（６）　命令到着待ち状態の場合

：この状態は２つある。

（ａ）　命令フェッチ要求が出されても何らかの原因（キャッシュミス等）により命令供給が行なわれない場合

：ＩＣ＿ｆｅｔｃｈ＿Ｌ＝１かつＩＣ＿ｒｅａｄｙ＝０

この状態では前のサイクルのｑｕｅｕｅ＿ｔｏｐが保持される。したがって、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿Ｌ

となる。

（ｂ）　たとえば命令メモリがアクセスを受入れられない状態にあるとき

：この状態はＩＣ＿ｍｉｓｓ＿Ｌ＝１かつＩＣ＿ｒｅａｄｙ＝０

この状態では、命令が供給されていないため、アドレスｑｕｅｕｅ＿ｔｏｐは前の値を保持する。したがって、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅ＿ｔｏｐ＿Ｌ

となる。

（７）　たとえばキャッシュミス等による命令到着待ち状態において命令が与えられ、かつキュー９に空き領域が存在する場合：

ＩＣ＿ｍｉｓｓ＿Ｌ＝１かつＩＣ＿ｒｅａｄｙ＝１かつｑｕｅｕｅ＿ａｖａｉｌ＝１

この状態では、与えられた命令をキュー９にすべて書込むことができる。このときには、このサイクルでは、ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿Ｌ＋４

となる。ただし「＋」はモジュル１２のモジュロ加算である。

（８）　命令到着待機状態において命令が供給されてもキュー９に空き領域が存在しない場合

：ＩＣ＿ｍｉｓｓ＿Ｌ＝１、かつＩＣ＿ｒｅａｄｙ＝１、かつｑｕｅｕｅ＿ａｖａｉｌ＝０

このとき、キュー９には命令を書込むことができないので、アドレスｑｕｅｕｅ＿ｔｏｐは前のサイクルの値を保持する。したがって、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿Ｌ

となる。

（８）　命令フェッチ要求を出しておらず、また命令到着待ち合わせ状態でもない場合

：ＩＣ＿ｆｅｔｃｈ＿Ｌ＝０かつＩＣ＿ｍｉｓｓ＿Ｌ＝０

この状態はたとえば割込み処理などにより、命令メモリの命令の実行が中断される状態である。この状態ではアドレスｑｕｅｕｅ＿ｔｏｐは前のサイクルの値を維持する。したがって、

ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＝ｑｕｅｕｅ＿ｔｏｐ＿Ｌ

となる。この上述の論理を第６図に一覧にして示す。

このキュートップロジック３０が実行する論理は上述の説明から明らかであろう。このキュートップロジック３０の具体的構成は、第６図に示す論理表を満足する構成であればどのようなものであってもよい。たとえば単純には、この第６図に示す表の各列（横方向）が示すロジックをたとえばＡＮＤゲートにより形成し、この各列のロジックの論理和をとる回路構成を用いて実現できる。

次にスコープネクストロジック３１の実行する論理動作について説明する。

（１）　リセット状態：信号ｒｅｓｅｔ＝１

この状態ではアドレスｓｃｏｐｅは０に設定される。すなわち、

ｓｃｏｐｅ＿ｎｅｘｔ＝０である。

（２）　分岐命令の発行

：ｂｒ＿ｔａｋｅｎ＝１

この場合、第４図に示すように、アドレスｓｃｏｐｅは分岐命令を格納するキュー９のアドレスに設定される。したがって、ｓｃｏｐｅ＿ｎｅｘｔ＝ｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙ

となる。

（３）　この上述の状態以外の場合：ｒｅｓｅｔ＝０かつｂｒ＿ｔａｋｅｎ＝０

この状態においては、アドレスｓｃｏｐｅは発行命令数に従ってシフトする。すなわち、

ｓｃｏｐｅ＿ｎｅｘｔ＝ｓｃｏｐｅ＿Ｌ＋ｉｓｓｕｅｄ＿ｉｎｓｔ＿ｃｏｕｎｔ

となる。ここで、「＋」はモジュール１２のモジュロ加算である。したがって、

ｓｃｏｐｅ＞１１＝＝１のとき、

ｓｃｏｐｅ＿ｎｅｘｔ＝ｓｃｏｐｅ＿ｎｅｘｔ－１２

となる。

このスコープネクストロジック３１の実現する論理は第７図に一覧にして示される。

ここでフラグｓｃｏｐｅ＞１１，ｑｕｅｕｅ＿ｔｏｐ＞１１はアドレスｓｃｏｐｅおよびｑｕｅｕｅ＿ｔｏｐがキュー９を一巡りしたことを示す。

次に第８図を参照して命令フェッチロジック３２の実現する論理について説明する。

（１）　命令フェッチ要求は、リセット状態のときまたは命令メモリが、命令フェッチ要求を受入れられる状態のとき発生される。すなわち、

ｒｅｓｅｔｌ＝１かつＩＣ＿ｂｕｓｙｌ＝１のときに

ＩＣ＿ｆｅｔｃｈ＝１

となる。ここで「ｌ」は否定を示す符号である。

（２）　命令フェッチ要求が出されないのは上述のごとくリセット状態または命令メモリが命令フェッチ要求を受入れない状態のときである。したがって上の状態（１）以外すべて

ＩＣ＿ｆｅｔｃｈ＝０

ｄｙ＝０

この状態では、次のサイクルは命令供給待機状態であり、次のサイクルで命令が与えられればその命令を書込む必要がある。フラグＩＣ＿ｍｉｓｓは１となる。

（４）　命令メモリから命令供給がされた場合

：ＩＣ＿ｒｅａｄｙ＝１

この状態は命令メモリから命令が供給されたことを示しており、次のサイクルにおいて命令待ち合わせ状態となる必要はなく、フラグＩＣ＿ｍｉｓｓは０となる。

（５）　動作状態において、分岐命令が発生せず、命令フェッチ要求も発生されず、また命令メモリからの命令供給指示も与えられない状態

：ｂｒ＿ｔａｋｅｎ＝０、かつＩＣ＿ｆｅｔｃｈ＿Ｌ＝０、かつＩＣ＿ｒｅａｄｙ＝０

この状態は、たとえば割込み処理などによる実行の中断状態などが生じており、フラグＩＣ＿ｍｉｓｓは前のサイクルの値を保持する必要があるため、

となる。次に命令ミスロジック３３が実現する論理について第９図を参照して説明する。

信号ＩＣ＿ｍｉｓｓは、命令メモリからの命令到着待ち状態であることを示すフラグである。この命令ミスロジック３３は次のサイクルのフラグＩＣ＿ｍｉｓｓの状態を決定する。

（１）　リセット状態：ｒｅｓｅｔ＝１

この状態においては、まだ何ら命令は発生されておらず命令を持つ必要もなく、フラグＩＣ＿ｍｉｓｓは０である。

（２）　動作時において、分岐命令が発生した場合

：ｂｒ＿ｔａｋｅｎ＝１

この状態においては、次のサイクルで分岐先命令のフェッチが行なわれるため、命令待ち合わせ状態となることはなく、フラグＩＣ＿ｍｉｓｓは０となる。

（３）命令フェッチ要求が出されても命令メモリから命令が供給されない場合

：ＩＣ＿ｆｅｔｃｈ＿Ｌ＝１かつＩＣ＿ｒｅａ

ＩＣ＿ｍｉｓｓ＝ＩＣ＿ｍｉｓｓ＿Ｌ、

となる。

次に第１０Ａ図および第１０Ｂ図を参照してキューアベールロジック３４が実現する論理動作について説明する。

（１）　リセット状態：ｒｅｓｅｔ＝１

この状態においては、キュー９においては命令は何ら格納されていないため、キュー９には、アドレスｑｕｅｕｅ＿ｔｏｐからの連続する４つのアドレス領域に４つの命令を書込むことができる。したがってこの場合、

ｑｕｅｕｅ＿ａｖａｉｌ＝１

となる。

（２）　分岐命令が発生した場合：ｂｒ＿ｔａｋｅｎ＝１

この状態においては、第４図に示すように、ｑｕｅｕｅ＿ｔｏｐから始まるアドレスに分岐先命令を書込む必要があるため、空き領域が等価的に存在することになる。したがって、

ｑｕｅｕｅ＿ａｖａｉｌ＝１

となる。

分岐が発生していない状態において、キューフルロジック（これは以下に説明する）がキュー９に空き領域（４エントリ分）が存在していることを示している場合には、

ｑｕｅｕｅ＿ａｖａｉｌ＝１となり、またこのキューフルロジックがキュー９が空き状態（４命令を書込むエントリ分の領域）にないことを示している場合に

ｑｕｅｕｅ＿ａｖａｉｌ＝０となる。すなわち、

この状態においてはｑｕｅｕｅ＿ａｖａｉｌ＝！ｑｕｅｕｅ＿ｆｕｌｌとなる。

次に、キューフルロジックについて説明する。このキューフルロジックは第１図におけるキューアベールロジック３４に含まれている。このキューフルロジックが実行する論理を第１０Ｂ図を参照して説明する。

（１）　このキューフルロジックはスコープネクストロジック３１からのアドレスｓｃｏｐｅとキュートップロジック３０からのアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎに従ってキュー９が空き領域を有しているか否かを判定する。ここで、スコープネクストロジック３１から出力されるアドレスｓｃｏｐｅは次のサイクルにおける読出先頭アドレスを示しているためｓｃｏｐｅ＿ｎｅｘｔとして説明する。

（１）　次のサイクルにおける読出先頭アドレスｓｃｏｐｅ＿ｎｅｘｔが次のサイクルにおける書込先頭アドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎよりも小さい場合

この場合、第１０Ｃ図に示すように、命令がｑｕｅｕｅ＿ｔｏｐ＿ｉｎから書込まれるため、アドレスｓｃｏｐｅ＿ｎｅｘｔとキュー９の全エントリ数との和からアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎとの差が４以上あれば次のサイクルで命令を書込むことができる。したがって、

ｓｃｏｐｅ＿ｎｅｘｔ＋１２－ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ≧４

の場合にはｑｕｅｕｅ＿ｆｕｌｌ＝ＦＡＬＳＥ（０；空き領域有）

となる。そうでない場合には、

ｑｕｅｕｅ＿ｆｕｌｌ＝ＴＲＵＥ（１；空き領域無し）

となる。

（２）　ｓｃｏｐｅ＿ｎｅｘｔ＞ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ：

この状態は第１０Ｄ図に示す状態である。この状態では、ｓｃｏｐｅ＿ｎｅｘｔとｑｕｅｕｅ＿ｔｏｐ＿ｉｎとの差Ｃが４以上であればキュー９に命令を書込むことができる。したがって、

ｓｃｏｐｅ＿ｎｅｘｔ－ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ≧４ならば、

ｑｕｅｕｅ＿ｆｕｌｌ＝ＦＡＬＳＥ

となる。

また、ｓｃｏｐｅ＿ｎｅｘｔ－ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ＜４ならば、

ｑｕｅｕｅ＿ｆｕｌｌ＝ＴＲＵＥ

となる。

（３）　ｓｃｏｐｅ＿ｎｅｘｔ＝ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ

この状態は次のサイクルにおける読出先頭アドレスと書込先頭アドレスとが等しい状態である。このとき、後述のキューステイト（ｑｕｅｕｅ＿ｓｔａｔｅ）に従って、キュー９の空き領域があるか否かの判別が行なわれる。すなわち、

ｑｕｅｕｅ＿ｓｔａｔｅ＝ＮＯＲＭＡＬならば、

ｑｕｅｕｅ＿ｆｕｌｌ＝ＦＡＬＳＥ

となる。また、ｑｕｅｕｅ＿ｓｔａｔｅ！＝ＮＯＲＭＡＬならば、

ｑｕｅｕｅ＿ｆｕｌｌ＝ＴＲＵＥ

となる。

次にキュー９の空き領域の有無を判定するための信号ｑｕｅｕｅ＿ｓｔａｔｅについて説明する。ここで、空き領域とは一度に命令メモリからフェッチされた複数の命令をすべて書込むことのできる領域であり、本実施例では最小４エントリの領域を示す。キュー９の状態は第１の状態の他に、第２の状態および第３の状態がある。まず、第１１Ａ図および第１１Ｂ図を参照して第２の状態（ＮＯＲＭＡＬ状態）について説明する。

第１１Ａ図および第１１Ｂ図は、キュー９の第２の状態（ＮＯＲＭＡＬ状態）のキュー９のアドレスｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅの位置関係を示す図である。第１１Ａ図において、アドレスｑｕｅｕｅ＿ｔｏｐは、キュー９のアドレス６を示し、アドレスｓｃｏｐｅはキュー９のアドレス２を示す。この状態は、

ｓｃｏｐｅ＜ｑｕｅｕｅ＿ｔｏｐ

と表わせる。

第１１Ｂ図において、命令が供給されず、命令の発行のみが行なわれた状態を示す。この場合、次のサイクルにおけるアドレスｓｃｏｐｅ＿ｎｅｘｔおよびｑｕｅｕｅ＿ｔｏｐ＿ｉｎは互いに等しくなる。この状態は、第２の状態（ＮＯＲＭＡＬ状態）において生じたものであり、この状態も第２の状態と称される。この状態は、

ｓｃｏｐｅ＿ｎｅｘｔ＝ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ

と表わせられる。

したがって、キュー９の第２の状態（ＮＯＲＭＡＬ状態）にある条件は、

ｓｃｏｐｅ≦ｑｕｅｕｅ＿ｔｏｐ

と表わせる。次に、第１２Ａ図および第１２Ｂ図を参照して、キュー９の第３の状態（ＲＥＶＥＲＳＥ状態）について説明する。

第１２Ａ図および第１２Ｂ図はキュー９の第３の状態（ＲＥＶＥＲＳＥ状態）におけるアドレスｓｃｏｐｅおよびｑｕｅｕｅ＿ｔｏｐの位置関係を示す図である。第１２Ａ図において、アドレスｓｃｏｐｅはキュー９のアドレス６を示し、アドレスｑｕｅｕｅ＿ｔｏｐは、キュー９のアドレス２を示す。この状態は、

ｓｃｏｐｅ＞ｑｕｅｕｅ＿ｔｏｐ

と表わせる。

この状態において、命令発行が行なわれず、命令供給が行なわれた状態を考える。すなわち、第１２Ａ図においてキュー９のアドレス２から命令が供給された場合を考える。この場合、第１２Ｂ図に示すように、次のサイクルにおけるアドレスは、

ｓｃｏｐｅ＿ｎｅｘｔ＝ｑｕｅｕｅ＿ｔｏｐ＿ｉｎ

となる。この状態では、未発行の命令のみでキュー９が満たされている。この状態では、次のサイクルにおける命令書込を禁止する必要がある。この状態をも合わせて第３の状態（ＲＥＶＥＲＳＥ状態）とよぶ。したがって、第３の状態（ＲＥＶＥＲＳＥ状態）の条件は、

ｓｃｏｐｅ≧ｑｕｅｕｅ＿ｔｏｐ

と表わせる。このキュー９の状態を信号ｑｕｅｕｅ＿ｓｔａｔｅによりモニタすることにより、前述のロジックｑｕｅｕｅ＿ｆｕｌｌの論理動作を通して、キュー９に空き領域があるか否かの判別を行なうことができる。

このキューステイトロジック５０が実現する論理動作を第１３図に一覧にして示す。以下、キューステイトロジック５０の動作について説明する。

（１）　リセット時の初期状態においては、アドレスｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅはキュー９のアドレス０に設定される。この状態は、キュー９の第１の状態であるが、キュー９へ命令を書込むことができるため、信号ｑｕｅｕｅ＿ｓｔａｔｅは第２の状態に設定される。すなわち、

ｒｅｓｅｔ＝１ならば、

ｑｕｅｕｅ＿ｓｔａｔｅ＝ＮＯＲＭＡＬ

となる。

（２）　アドレスｑｕｅｕｅ＿ｔｏｐが、キュー９のアドレス領域を一巡し、かつアドレスｓｃｏｐｅがまだキュー９のアドレス領域を一巡しない場合：

このとき、第１３図の（２）に示すように、キュー９は第３の状態（ＲＥＶＥＲＳＥ状態）となる。したがって、

ｑｕｅｕｅ＿ｔｏｐ＞１１＝＝１、かつ

ｓｃｏｐｅ＞１１＝＝０のとき、

ｑｕｅｕｅ＿ｓｔａｔｅ＝ＲＥＶＥＲＳＥ

となる。

（３）　アドレスｓｃｏｐｅがキュー９のアドレス領域を一巡した場合：

この状態では、アドレスｓｃｏｐｅ＿ｎｅｘｔ

がアドレスqueue_top_inを超えることはないので、キュー9は第2の状態（NORMAL状態）となる。したがって、

scope>11==1ならば、

queue_state=NORMAL

となる。

（4） 分岐が発生した場合：

この状態においては、分岐命令から始まるキュー9のアドレス領域に次の分岐先命令を格納する必要がある。したがってこの状態ではアドレスscope_nextおよびqueue_top_inは等しくされるが、命令の書込を可能とするために、キュー9の状態は第2の状態（NORMAL状態）となる。したがって、

br_taken=1ならば、

queue_state=NORMAL

となる。次に、キュー9の「有効性フラグ」部分に書込まれるフラグを決定する命令アベールロジック35が実現する論理について第14図を参照して説明する。

第14図は命令アベールロジック35が実現する論理を一覧にして示す図である。

（1） 初期状態：reset=1

この状態では、まだキュー9に有効な命令は何ら格納されていないため、有効性フラグはオフ（0）にされる。したがって、

reset=1ならば、

inst_avail_in=0

となる。

（2） 分岐が発生した場合、

この状態では、分岐命令が格納されたアドレスから続く4エントリの命令の発行を禁止する必要がある。したがってこの場合には有効性フラグはオフ（0）とする。したがって、

br_taken=1、ならば、

inst_avail_in=0

となる。

（3） 命令フェッチ要求が出された場合：

この状態では、アドレスqueue_topから始まる命令の有効／無効は命令メモリからの命令の供給の有無に対応する。したがって、

IC_fetch_L=1、ならば、

inst_avail_in=IC_ready

となる。

（4） 命令到着待ち状態の場合：

この状態は、そのサイクルでの命令フェッチ要求の有無にかかわらず生じる。このときも、アドレスqueue_topから始まる命令の有効／無効は命令メモリからの命令供給の有無に対応する。したがって、

IC_miss_L=1、ならば、

inst_avail_in=IC_ready

となる。

（5） 上述の4状態のいずれの状態でもない場合：

この状態においては、アドレスqueue_topから始まる4エントリに書込まれる命令は要求する命令ではないため、有効性フラグはオフとされる。したがって、

else, inst_avail_in=0

となる。

第15図はキュー初期化ロジック36の実現する論理を一覧にして示す図である。以下、第15図を参照してキュー初期化ロジック36の論理動作について説明する。ここで、キューの初期状態（第1の状態）とは、キューのリセット状態、分岐命令が発生した状態、および命令待ち状態のいずれかの状態を示す。すなわち、キュー9に必要とされる命令が書込まれていない状態を表わしている。

（1） リセット状態：

この状態においては、まだキュー9には何ら命令は書込まれていないため。キュー9は初期状態にある。したがって、

reset=1、ならば

queue_init_state=1

となる。

（2） 分岐が発生した場合：

この状態においては、キュー９は分岐先命令を格納する必要があるが、この命令は、まだキュー９に書込まれていないため、キュー９は初期状態にある。したがって、

ｂｒ＿ｔａｋｅｎ＝１、ならば、

ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＝１、

となる。

（３）　命令フェッチ要求が出されても命令メモリから命令が供給されない場合：

この状態においてはキュー９は命令待ち状態となるため、キュー９は初期状態となる。したがって、

ＩＣ＿ｆｅｔｃｈ＿Ｌ＝１、かつ

ＩＣ＿ｒｅａｄｙ＝０、ならば、

ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＝１、

となる。

（４）　命令フェッチ要求が出されかつ命令メモリから命令が供給された場合：

この状態においてはキュー９には所望の必要とされる命令が書込まれたことにより、キュー９は初期状態ではない。したがって、

ＩＣ＿ｆｅｔｃｈ＿Ｌ＝１、かつ

ＩＣ＿ｒｅａｄｙ＝１、ならば、

ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＝０、

となる。

（５）　キュー９が命令待ち状態にあり、かつ命令メモリから命令が供給されない場合：

この状態では、キュー９はフェッチ要求した命令の到着を待っている。したがって、キュー９は初期状態にある。したがって、

ＩＣ＿ｍｉｓｓ＿Ｌ＝１、かつ

ＩＣ＿ｒｅａｄｙ＝０、ならば

ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＝１

となる。

（６）　キュー９が命令待ち状態にありかつ命令メモリが命令を供給した場合：

この状態では、キュー９にはフェッチ要求した命令が書込まれたため、キュー９は初期状態ではなくなる。したがって、

ＩＣ＿ｍｉｓｓ＿Ｌ＝１、かつ

ＩＣ＿ｒｅａｄｙ＝１ならば、

ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＝０

となる。

（７）　リセット状態になく、分岐発生状態でもなく、命令フェッチ要求を行なっておらず、また命令待ち合わせ状態でもない場合：

この状態においては、キュー９は命令メモリからの命令の供給の有無にかかわらず、キュー９は次のサイクルの状態では、そのときの状態を保持すべき状態にあり、またすなわち、第６図に示すようにアドレスｑｕｅｕｅ＿ｔｏｐは同一アドレスを続いて示す必要がある。したがって、

ｒｅｓｅｔ＝０、かつｂｒ＿ｔａｋｅｎ＝０、かつ

ＩＣ＿ｆｅｔｃｈ＿Ｌ＝０かつＩＣ＿ｍｉｓｓ＿Ｌ＝０、ならば、

ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＝ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅ＿Ｌ

となる。

なおロジック３０ないし３６および５０が実現する論理を一欄にして示しているが、これらの論理を実現するための具体的構成は当業者であれば容易に作成することができるであろう。

命令デコーダ２６は、Ｌラッチ回路１４～１６から与えられるアドレスＩＤ＿ＰＣ＿Ｌ、命令ＩＤ＿ＩＲ＿Ｌおよび有効性フラグＩＤ＿ｉｎｓｔ＿ａｖａｉｌ＿Ｌをデコードし、並列処理可能な命令を検出し、該命令を対応の機能ユニットへ転送するとともに、信号ｂｒ＿ｔａｋｅｎ、データｉｓｓｕｅｄ＿ｉｎｓｔ＿ｃｏｕｎｔおよびｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙを発生する。

この第１図に示す回路構成は１サイクルですべて所望の動作を実行する。分岐発生時において、分岐命令と同時に命令デコーダ２６に与えられた命令のうち分岐命令以後の命令については、デコード動作を行なっても機能ユニットへの発行を停止する構成であってもよく、また機能ユニットへその命令を与える一方そのサイクルにおける機能ユニットの動作を禁止する構成であってもよい。

次に、各ロジック３０ないし３６および５０の動

作について図面を参照して説明する。

第１６図は第３図に示すキュー、ｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅの基本的な動きに対応する各ロジックの動作を示す信号波形図である。第１６図には、各サイクルと第３図に示す各ステップとの対応関係を合わせて示す。また、各サイクルの初めはクロック信号Ｔにより決定される。この信号波形図においてサイクルの中間でロジックの出力の変化は、クロック信号Ｌに応答して生じている。

（１）　サイクル０および１：

このサイクルはリセット信号ｒｅｓｅｔが"１"にあり、キュー９は初期状態にある。したがって、信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅがオン（１）、アドレスｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅは０であり、また有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎは０である。

また、リセット状態においては、命令メモリに対するフェッチ要求は行なわれないため、命令フェッチ要求フラグＩＣ＿ｆｅｔｃｈはオフ（０）である。また、キュー９はリセット状態においては空き領域が十分に存在するため、信号ｑｕｅｕｅ＿ａｖａｉｌは空き領域があることを示すため"１"である。

（２）　サイクル２：

このサイクル２においてリセット信号ｒｅｓｅｔが"０"となり、リセットが解除される。このリセットの解除はサイクルの中途で行なわれており、各ロジックの出力状態はサイクル１と同様である。

（３）　サイクル３：

このサイクルにおいては、命令フェッチロジック３２からの命令フェッチ要求ＩＣ＿ｆｅｔｃｈをオン状態とする。これにより命令メモリに対する命令フェッチ要求が行なわれる。この発行された命令フェッチ要求に対して命令メモリから命令が供給される。命令メモリからは命令供給を示す信号ＩＣ＿ｒｅａｄｙがオン状態となる。このオン状態のＩＣ＿ｒｅａｄｙに応答して、命令アベールロジック３５からの出力される有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎがオン状態となる。このとき、キュー初期状態フラグｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅはオン状態にあり、キュー９が初期状態にあることを示している。

（４）　サイクル４：

信号ＩＣ＿ｆｅｔｃｈおよびＩＣ＿ｒｅａｄｙがともにオン状態となったことにより、キュー９の初期状態が解除され、信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅがオフ（０）とされ、キュー９に対する命令等の書込が実行される。すなわち、アドレスｑｕｅｕｅ＿ｔｏｐが示すアドレス０からの領域に命令メモリから供給された命令ＩＣ＿ｄａｔａおよび命令フェッチステージから伝達された各命令の論理アドレスＩＦ＿ＰＣ＿Ｌおよび命令アベールロジック３５からの有効性フラグ部ｉｎｓｔ＿ａｖａｉｌ＿ｉｎが書込まれる。

またこのときアドレスｓｃｏｐｅから始まるアドレス０に書込まれた命令およびアドレスおよび有効性フラグが読出され、ラッチ回路１４ないし１６へ与えられる。このラッチ回路１４ないし１６へ与えられたデータはクロック信号Ｌに応答して命令デコーダ２６へ与えられてそこで解読される。その結果、サイクル４において２つの命令が並列処理可能であるとして対応の機能ブロックに発行される。これにより、命令デコーダ２６からは発行命令数が２であることを示す信号ｉｓｓｕｅｄ＿ｉｎｓｔ＿ｃｏｕｎｔが発生され、スコープネクストロジック３１へ与えられる。アドレスｓｃｏｐｅ＿ｎｅｘｔは２となる。また、アドレスｑｕｅｕｅ＿ｔｏｐは４を加算され、４となる。

（５）　サイクル５：

このサイクルにおいては、アドレスｑｕｅｕｅ＿ｔｏｐはアドレス４を示しており、またアドレスｓｃｏｐｅはアドレス２を示している。サイクル４と同様に命令メモリから命令が供給され、アドレス４ないし７に命令が書込まれる。このとき同時に、キュー９からアドレス２からの命令が読出され、ラッチ回路１４ないし１６を介して命令デコーダ２６へ与えられる。命令デコーダ２６からは１つの命令が機能ユニットへ発行される。し

たがって、スコープネクストロジック３１から出力されるアドレスｓｃｏｐｅは３となる。また、キュートップロジック３０から出力されるアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎは８となる。

（６）　サイクル６：

このサイクル６においてもサイクル４と同様に命令メモリから命令が供給される。これにより、キュー９のアドレス８ないし１１に命令が書込まれる。

一方、命令デコーダ２６からは機能ユニットへ発行される命令が存在しなかったため、発行命令数０を示す信号ｉｓｓｕｅｄ＿ｉｎｓｔ＿ｃｏｕｎｔが発生されスコープネクストロジック３１へ与えられる。

キュートップロジック３０からの次のサイクルのアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎはアドレス０を示す。スコープネクストロジック３１から出力されるアドレスｓｃｏｐｅ＿ｎｅｘｔは３である。このとき、キュー９のアドレス３には未発行の命令が残る。したがって、キューアベールロジック３４は、そこに含まれるキューフルロジックの機能によりキュー９に空き領域がないことを示すために、信号ｑｕｅｕｅ＿ａｖａｉｌをオフ（０）に設定する。これにより、次のサイクルにおけるキュー９への命令の書込が禁止される。

（７）　サイクル７：

このサイクルにおいては、キュー９のアドレス３から命令が読出され、命令デコーダ２６から３つの命令が発行される。これにより、スコープネクストロジック３１から出力されるアドレスｓｃｏｐｅ＿ｎｅｘｔは６となる。アドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎは、そのとき信号ｑｕｅｕｅ＿ａｖａｉｌがオフ状態であったため変更せず０である。このとき、アドレスｓｃｏｐｅ＿ｎｅｘｔが６であり、一方アドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎが０であり、キュー９に空き領域が生じたため、キューアベールロジック３４からの出力信号ｑｕｅｕｅ＿ａｖａｉｌはオンとなり、次のサイクルにおけるキュー９への命令の書込が許可される。

（８）　サイクル８：

アドレスｑｕｅｕｅ＿ｔｏｐはサイクル７と同様０であるが、キュー９は命令を書込むことができるため（ｑｕｅｕｅ＿ａｖａｉｌはオン状態）、キュー９のアドレス０ないし３の領域に命令が書込まれる。このとき、アドレスｓｃｏｐｅは６であり、キュー９はアドレス６を示しており、このキュー９のアドレス６からの命令が命令デコーダ２６へ与えられる。

なお、このとき、命令フェッチ要求ＩＣ＿ｆｅｔｃｈが連続して発行されるため、サイクル７とサイクル８とで命令メモリから与えられる命令の内容が異なることも考えられる。これは、キューアベールロジック３４からの信号ｑｕｅｕｅ＿ａｖａｉｌを命令フェッチステージへ与え、この信号ｑｕｅｕｅ＿ａｖａｉｌがオフの場合には、命令フェッチステージに含まれるプログラムカウンタ（ＩＦ＿ＰＣ）のカウンタ動作を禁止し、サイクル７とサイクル８とで命令メモリの同一の内容が与えられるように構成すれば、同一命令を命令メモリから与えることができる。

なお、各ロジック３０ないし３６および５０は与えられた信号を論理処理しているだけであり、クロック信号ＴまたはＬとの同期動作は行なってはいない。

第１７図は、第４図に示す分岐発生時における各ロジックの動作を示す信号波形図である。第１７図において第４図の各ステップが各サイクルに対応づけて示される。以下、第１７図を参照して分岐発生時におけるキュー、ｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅの動きについて説明する。

（１）　サイクル０：

アドレスｑｕｅｕｅ＿ｔｏｐは１０であり、アドレスｓｃｏｐｅは２である。したがってキュー９からは命令８、９、１０および１１がアドレス１０、１１、０および１に書込まれる。一方、キュー９のアドレス２ないし５の命令０ないし３が読出される。このサイクル０において、キュー９のアドレス４に格納されていた分岐命令２３により分岐が発生した場合を想定する。この場合、信

号ｂｒ＿ｔａｋｅｎがオン（１）となる。この分岐が生じたことにより、命令アベールロジック３５からの有効フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎはオフ（０）となる。

また、この分岐発生により、キュートップロジック３０から出力されるアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎは分岐命令２が格納されていたアドレスｂｒａｎｃｈ＿ｉｎｓｔ＿ｅｎｔｒｙに従って４となり、また同様に、スコープネクストロジック３１から出力されるアドレスｓｃｏｐｅ＿ｎｅｘｔも４となる。

またキュー初期化状態ロジック３６の出力ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅもこの分岐発生により初期状態（第１の状態）を示すために１となる。また、キューアベールロジック３４の出力へはこの分岐発生によりキューステイトロジック５０からＮＯＲＭＡＬ状態を示す信号が与えられ、信号ｑｕｅｕｅ＿ａｖａｉｌは１のままである。

（２）　サイクル１：

このサイクルにおいては、アドレスｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅはともに４に設定され、またキュー９のアドレス４ないし７の有効性フラグに０が書込まれる。この分岐が発生した場合には、分岐先命令のアドレスがこのサイクルで命令フェッチステージから命令メモリへ与えられ、分岐先命令はその次のサイクルに命令メモリから供給される。したがって、この分岐先命令はこのサイクルではキュー９には書込まれない。

このサイクル１においては、アドレス４から始まる命令は命令デコーダ２６へ与えられるが、この命令内容は機能ユニットへ発行／実行してはならないものである。このため、命令デコーダ２６は、関連の有効性フラグｉｎｓｔ＿ａｖａｉｌがオフ状態となるためその発行を行なわない。

さらに、キュー初期化ロジック３６からの出力信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅはオン（１）であるため、キュートップロジック３０およびスコープネクストロジック３１から出力されるアドレスｑｕｅｕｅ＿ｔｏｐ＿ｉｎおよびｓｃｏｐｅ＿ｎｅｘｔは４を保持する。またこのサイクルにおいては分岐は生じないため、命令デコーダ２６からの信号ｂｒ＿ｔａｋｅｎはオフとなる。これにより、命令アベールロジック３５からの有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎはオンとなる。

（３）　サイクル２：

このサイクルにおいては、分岐先命令が供給され、キュー９に格納されるとともに、関連の有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎもオンとなる。このときアドレス４から始まる命令すなわち分岐先命令が命令デコーダ２６へ与えられデコードされる。このサイクルにおいては、信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅはオフとされているため、その処理内容に応じてアドレスｑｕｅｕｅ＿ｔｏｐおよびｓｃｏｐｅはそれぞれ変更される。ここで、サイクル２においては、信号ＩＣ＿ｒｅａｄｙがオンのため、サイクル２において分岐先命令が供給されていることを示している。

第１８図は第５図に示す動作状態における各ロジックの動作を示す信号波形図である。以下、第１８図を参照して、発生された命令フェッチ要求に対し命令メモリから命令が供給されない場合の動作について説明する。なお、第１８図において第５図に示すステップも合わせて各サイクルに対応づけて示されている。

（１）　サイクル０：

このサイクルにおいてアドレスｑｕｅｕｅ＿ｔｏｐが０、アドレスｓｃｏｐｅが６である。このサイクル０において発生された命令フェッチ要求（ＩＣ＿ｆｅｔｃｈがオン）に対し、命令メモリから命令が供給されない場合を想定する。このとき、命令メモリからの信号ＩＣ＿ｒｅａｄｙはオフとなり、また命令メモリへのアクセスを禁止するため、信号ＩＣ＿ｂｕｓｙはオンとなる。このオフ状態の信号ＩＣ＿ｒｅａｄｙに応答して有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎがオンからオフ状態へ変わる。

（２）　サイクル１：

このサイクルにおいては、サイクル０において、ＩＣ＿ｆｅｔｃｈが１でありかつＩＣ＿ｒｅａｄ

yが0のため、信号ＩＣ＿ｍｉｓｓが1となり、キュー9が命令待ち状態にあることを示す。またキュー初期化ロジック３６からの出力ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅは、サイクル0において、命令要求が発生されても命令供給がされなかったことに応答して"1"となりキュー9が初期状態（第1の状態であり命令待ち状態）に設定され、アドレスｑｕｅｕｅ＿ｔｏｐの変更が禁止される。このサイクルにおいては、アドレス4から始まるキュー9の領域にはオフ（0）の有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎが書込まれ、該領域に格納された命令の発行／実行が禁止される。

一方、アドレスｓｃｏｐｅが示すアドレスキューからの命令は命令デコーダ２６へ与えられ、2つの命令が発行され、次のサイクルにおけるアドレスｓｃｏｐｅ＿ｎｅｘｔは１１に変更される。

（3）　サイクル2：

この状態においても待ち状態であり、ＩＣ＿ｆｅｔｃｈはオフ、信号ＩＣ＿ｍｉｓｓはオン、および信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅはオン状態である。この状態においては、アドレス4から始まる4エントリ領域に命令無効を示すフラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎが書込まれる。

一方、アドレスｓｃｏｐｅから始まる命令は読出され、命令デコーダ２６へ与えられ、2つの命令が発行され、次のサイクルにおける書込先頭アドレスｓｃｏｐｅ＿ｎｅｘｔは1に変更される。

このサイクル2において、命令メモリから命令が供給されたとする。このとき、信号ＩＣ＿ｒｅａｄｙがオン状態となり、信号ＩＣ＿ｂｕｓｙはオフ状態となる。これに応答して、有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎはオン状態へ移行する。この状態により、キュー9は命令待ち状態が解除される。

（4）　サイクル3：このサイクルにおいては、命令メモリから到達した命令がアドレスｑｕｅｕｅ＿ｔｏｐが示す4から始まる領域に登録され、またその対応の有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎも1に設定される。このとき、アドレスｓｃｏｐｅが示すアドレス位置からの命令に対する命令デコーダ２６による機能ユニットへの発行が行なわれる。このサイクル3においてはキュー9は初期状態から解除されたため、初期化状態信号ｑｕｅｕｅ＿ｉｎｉｔ＿ｓｔａｔｅはオフとされ、アドレスｑｕｅｕｅ＿ｔｏｐの変更が再開される。また、信号ＩＣ＿ｆｅｔｃｈもオン状態とされ、命令フェッチ要求も再開される。

上述のように、複数の命令をそのアドレスおよび有効性フラグとともに同時に書込かつ読出を行なうことのできるキューを設け、このキューの書込／読出ポインタを、分岐発生、命令供給状態およびリセット状態の第1の状態のいずれにあるかおよびキューに空き領域があるか否かに従ってポインタの制御および命令の機能ユニットへの発行可否を制御するように構成したため、命令デコーダへ効率的に命令を供給することができ、高速で命令を実行することのできるスーパースカラーを得ることができる。次に、この複数の命令を同時に書込／読出することのできるキュー9の構成および動作について説明する。

第１９図は命令メモリ1からフェッチされる命令とキュー9へ書込まれる命令との対応関係を概念的に示す図である。まず、第１９図を参照して、キュー9へのアドレスＩＦ＿ＰＣ＿Ｌ、命令ＩＣ＿ｄａｔａおよび有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎの書込動作について説明する。

命令フェッチステージ2からは命令フェッチ要求ＩＣ＿ｆｅｔｃｈに応答して、命令メモリ1のアドレスＩＦ＿ＰＣが発生されて命令メモリ1へ与えられる。命令メモリ1からはこのアドレスＩＦ＿ＰＣに従って4つの命令ＩＣ＿1、ＩＣ＿2、ＩＣ＿3およびＩＣ＿4が同時に読出される。この4つの命令ＩＣ＿1～ＩＣ＿4は並列に命令ＩＣ＿ｄａｔａとしてキュー9の命令領域へ与えられる。

一方、命令フェッチステージ2からはこのアドレスＩＦ＿ＰＣ＿Ｌがキュー9へ供給される。このアドレスＩＦ＿ＰＣ＿Ｌは4つの命令ＩＣ＿1～ＩＣ＿4の各アドレスを示しており、これらのアドレスが並列にキュー9のアドレス領域へ与え

られる。この命令は任意の長さであってもよいが、キュー9のエントリの効率的利用からは命令メモリ1からの命令の長さはすべて同一の長さに設定される。

命令メモリ1から発生される信号IC_busyは第1図に示す命令フェッチロジック32へ与えられる。命令メモリ1から発生される信号IC_readyは命令アベールロジック35へ与えられる。この命令アベールロジック35へ与えられる他のロジックからの信号は図示していない。

キュー9は、これらのアドレスIF_PC_L、命令IC_dataおよび有効性フラグinst_avail_inを受け、書込イネーブル信号wen（これは第1図のキューアベールロジック34から発生される）に応答して、アドレスqueue_topが示すアドレスから始まる4つのエントリ領域へ同時にこれらを書込む。

このときまた、アドレスscopeから始まる4つのエントリの内容が読出される。このアドレスIF_PC_Lは、クロック信号Lに応答して確定状態になるものではあるが、クロック信号Tに応答して発生されるものであってもよい。

キュー9に書込まれた命令およびアドレスが同時にまた読出される場合もあり、このキュー9から読出された命令およびアドレスは、クロック信号Lに応答して保持動作を実行するラッチ回路14および15で保持されるため、このキュー9の書込データはラッチ回路14および15の保持動作前に確定していれば良い。したがって、このアドレスIF_PC_Lが同時にまた読出されるときにはこのラッチ回路のラッチタイミングを少し遅らせておけば、アドレスIF_PC_Lがクロック信号Lに応答して確定状態とされるものであっても何ら誤動作は生じない。

第20図はキュー9の命令領域へ与えられる命令IC_dataの配置を示す図である。第20図を参照して、同時に読出される4つの命令IC_1～IC_4の各々は、各ビットごとに集められる。すなわち、命令IC_dataの第0ビット領域には4つの命令IC_1～IC_4の第0ビットIC10～IC40が順次格納され、以下この順番で第32ビットまで同様に配置される。ここで、命令IC-1～IC-4はすべて32ビット構成の場合が一例として示されている。

第21図はキュー9のアドレス領域へ伝達される命令アドレスIF_PC_Lの配置を示す図であり、命令IC_dataと同様各命令ごとにビットが順番に配置される。

第22図は有効性フラグの配置を示す図である。この有効性フラグinst_avail_inは4エントリに対応して4ビット構成とされ、すべて同一の値をとる。

第23図はキューの全体の構成を概略的に示す図である。第23図においてキュー9は、アドレスqueue_topをデコードし、4エントリを選択するための信号を発生する書込デコーダWRDと、アドレスscopeをデコードし、命令を読出すべき4エントリを選択する信号を発生する読出デコーダRDDと、各々が1エントリ領域を構成し、アドレス、命令および有効性フラグを格納する並列に設けられたエントリ記憶装置92-0～92-11を含む。

書込デコーダWRDおよび読出デコーダRDDは、それぞれ与えられたアドレスqueue_topおよびscopeに応答してその変化に従ってデコード動作を実行する。すなわちこれらの書込デコーダWRDおよびRDDは非同期デコード動作をスタティックに行なっており、特にこれらの書込デコードおよびWRDおよび読出デコーダRDDの動作タイミングを規定する制御信号は用いられてはいない。

書込デコーダWRDには12本の書込ワード線WD0～WD11が接続され、アドレスqueue_topに従ってこれらの12本の書込ワード線WD0～WD11のうちの1本を選択状態とされる。読出デコーダRDDも12本の読出ワード線RD0～RD11を有しており、アドレスscopeに応答してこれらの12本の読出ワード線RD0～RD11のうちの1本を選択状態とする。後に詳細に説明するが、1つの書込ワード線WD

ｉ（ｉ＝０～１１）および１本の読出ワード線ＲＤｉにはそれぞれ連続して隣接する４つのエントリ記憶装置が接続される。したがって、１本の書込ワード線または読出ワード線が選択状態とされることにより、同時に４つのエントリ記憶装置が選択状態とされる。

エントリ記憶装置９２－０～９２－１１の各々は、データ書込ポートを選択するための書込アクセスゲートｗ０、ｗ１、ｗ２、およびｗ３と、データ読出経路を選択するための読出アクセスゲートｒ０、ｒ１、ｒ２およびｒ３を含む。この書込アクセスゲートｗｊ（ｊ＝０～３）および読出アクセスゲートｒ０～ｒ３のいずれか１つを選択することによりデータの書込／読出ポートが選択される。４つの隣接するエントリ記憶装置の書込アクセスゲートの異なる書込アクセスゲートが１本の書込ワード線ＷＤｉに接続される。同様に、４つの隣接するエントリ記憶装置の異なる読出アクセスゲートが１本の読出ワード線ＲＤｉに接続される。

たとえば、読出ワード線ＷＤ０には記憶装置９２－０の書込アクセスゲートｗ０、記憶装置９２－１の書込アクセスゲートｗ１、記憶装置９２－２の書込アクセスゲートｗ２、および記憶装置９２－３の書込アクセスゲートｗ３が接続される。また、読出ワード線ＲＤ０には、記憶装置９２－０の読出アクセスゲートｒ０、記憶装置９２－１の読出アクセスゲートｒ１、記憶装置９２－２の読出アクセスゲートｒ２および記憶装置９２－３の読出アクセスゲートｒ３が接続される。

エントリ記憶装置９２－０～９２－１１の各々は、書込データ入力ポートｉｗｉ、書込データ出力ポートｉｗｏ、読出データ入力ポートｉｒｉ、読出データ出力ポートｉｒｏ、書込アドレス入力ポートｐｗｉ、書込アドレス出力ポートｐｗｏ、読出アドレス出力ポートｐｒｏ、読出アドレス入力ポートｐｒｉ、書込有効性フラグ入力ポートａｗｉ、書込有効性フラグ出力ポートａｗｏ、読出有効性フラグ出力ポートａｒｏ、および読出有効性フラグ入力ポートａｒｉを含む。

記憶装置９２－０～９２－１１の書込命令入出力ポートｉｗｉおよびｉｗｏはすべて共通の命令書込ビット線ＩＷＢに接続され、読出命令入出力ポートｉｒｏおよびｉｒｉは共通に命令読出ビット線ＩＲＢへ接続される。

記憶装置９２－０～９２－１１の書込アドレス入出力ポートｐｗｉおよびｐｗｏはアドレス書込ビット線ＰＷＢに接続され、記憶装置９２－０～９２－１１の読出アドレス入出力ポートｐｒｏおよびｐｒｉはアドレス読出ビット線ＰＲＢに接続される。

記憶装置９２－０～９２－１１の書込有効性フラグ入出力ポートａｗｉおよびａｗｏは有効性フラグ書込ビット線ＡＷＢに接続され、読出有効性フラグ入出力ポートａｒｉおよびａｒｏは有効性フラグ読出ビット線ＡＲＢに接続される。

命令書込ビット線ＩＷＢ、命令読出ビット線ＩＲＢ、アドレス書込ビット線ＰＷＢ、およびアドレス読出ビット線ＰＲＢは４命令分のデータを伝達することのできるビット幅を有しており、本実施例においては、４・３２＝１２８のビット幅を有している。有効性フラグ書込ビット線ＡＷＢおよび有効性フラグ読出ビット線ＡＲＢは４ビットの幅を有している。これらの読出ビット線および書込ビット線上に４命令分の内容が同時に伝達される。

記憶装置９２－０～９２－１１の各々の入力および出力ポートはともに４つのポートを有しており、この４つのポートのうちの１つが書込アクセスゲートｗ０～ｗ３および読出アクセスゲートｒ０～ｒ３より選択される。したがって、同時に選択状態とされる４つの記憶装置は互いに異なる入力ポートおよび出力ポートが選択状態とされる。

情報の入出力を行なうために、第１図に示すキューアベールロジック３４から発生される書込イネーブル信号ｗｅｎに応答して所定のタイミングで書込パルスＷＰを発生する書込パルス発生器ＷＰＧと、書込パルス発生器ＷＰＧからの書込パルスＷＢに応答してオン状態となる、トライステートバッファＢＦ１、ＢＦ２およびＢＦ３が設けら

れる。このトライステートバッファＢＦ１～ＢＦ３は書込パルスＷＰが発生されない場合その出力状態をハイインピーダンス状態とする。

トライステートバッファＢＦ１は、命令メモリからの命令ＩＣ＿ｄａｔａを受け、命令書込ビット線ＩＷＢ上へ伝達する。トライステートバッファＢＦ２は、アドレスＩＦ＿ＰＣ＿Ｌをアドレス書込ビット線ＰＷＢ上へ伝達する。トライステートバッファＢＦ３は有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎを有効性フラグ書込ビット線ＡＷＢ上へ伝達する。命令読出ビット線ＩＲＢからは命令ｉｎｓｔが出力され、ラッチ回路１４へ伝達される。読出ビット線ＰＲＢからはアドレスＰＣが読出され、Ｌラッチ回路１５へ伝達される。読出ビット線ＡＲＢ上の有効性フラグｉｎｓｔ＿ａｖａｌ.ｌはＬラッチ回路１６へ接続される。

次にこの第２３図に示すキューの動作について簡単に説明する。

まずデータ書込動作について説明する。アドレスｑｕｅｕｅ＿ｔｏｐに従って、書込デコーダＷＲＤにより１本の書込ワード線ＷＤｉが選択状態とされる。これにより、連続して隣接する４つのエントリ記憶装置が選択されるとともにそれぞれの異なる入力ポートが選択状態とされる。これにより、４つのエントリ記憶装置へ同時に書込を行なうことが可能となる。次いで、キュー９に空領域が存在するとキューアベールロジック３４からの書込イネーブル信号ｗｅｎがオン状態になり、書込パルス発生器ＷＰＧから所定のタイミングで書込パルスＷＰが発生され、トライステートバッファＢＦ１ないしＢＦ３が導通状態とされ書込データが選択されたエントリ記憶装置の対応のポートへ伝達され、４エントリに命令、アドレスおよび有効性フラグが同時に書込まれる。

読出動作も同様であり、アドレスｓｃｏｐｅが与えられると読出デコーダＲＤＤは１本の読出ワード線ＲＤｉを選択し、４つのエントリ記憶装置の異なる読出アクセスゲートを選択する。これにより４つのエントリ記憶装置から同時に４つのエントリ内容を読出すことができる。この読出されたデータは読出ビット線ＩＲＢ、ＰＲＢおよびＡＲＢを介して対応のラッチ回路１４、１５および１６へ伝達される。

第２４図は１個のエントリ記憶装置の構成を示す図である。第２４図において１個のエントリ記憶装置９２を総称的に示す。命令記憶装置９３－１、アドレス記憶装置９３－２および有効性フラグ記憶装置９４を含む。命令記憶装置９３－１およびアドレス記憶装置９３－２は同一の構成を有し、書込ポート選択ゲートｗｂと、読出ポート選択ゲートｒｂと、書込アクセスゲートｗ０～ｗ３および読出アクセスゲートｒ０～ｒ３を含む。ゲートｗｂ，ｒｂの＜３：０＞は入出力ポートが、ともに０ないし３の４つあることを示しており、また＜３１：０＞は３２ビットのデータが、１つの選択されたポートから出力されることを示している。したがって、命令書込ビット線ＩＷＢおよび読出ビット線ＩＲＢ、アドレス書込ビット線ＰＷＢおよびアドレス読出ビット線ＰＲＢの１２８ビットの信号線のうち３２ビットの信号線が１つの記憶装置により使用される。どの３２ビットの信号線が使用されるかはアクセスゲートｗｂ０～ｗｂ３およびｒ０～ｒ３の選択により決定される。

有効性フラグ記憶装置９４は書込ポート選択ゲートｗｂおよび読出ポート選択ゲートｒｂと、書込アクセスゲートｗ０～ｗ３および読出アクセスゲートｒ０～ｒ３を含む。この書込ポート選択ゲートｗｂは、ポートを０ないし３の４つ有しており、１つの選択ポートから１ビットの有効性フラグが入出力される。すなわち、アクセスゲートｗ０～ｗ３およびｒ０～ｒ３を選択することにより、フラグ書込ビット線およびフラグ読出ビット線ＡＷＢおよびＡＲＢの４ビットのうち１ビットが、１つの有効性フラグ記憶装置９４により使用される。

第２５図は命令記憶装置９３－１およびアドレス記憶装置９３－２の構成をより詳細に示す図である。この記憶装置９３－１および９３－２は同一の構成を有しており、記憶装置９３を総称的に示す。第２５図を参照して、命令記憶装置９３－

1およびアドレス記憶装置93－2はともに、32ビットのビット記憶装置95－0～95－31を含む。このビット記憶装置95－0～95－31の各々は、命令またはアドレスの第0ビットないし第31ビットをそれぞれ記憶する。ビット記憶装置95－0～95－31の各々は、書込ポート選択ゲートwb0、wb1、wb2およびwb3と、読出ポート選択ゲートrb0、rb1、rb2およびrb3と、書込アクセスゲートw0～w3および読出アクセスゲートr0～r3を含む。このゲートwb0～wb3およびrb0～rb3の選択がアクセスゲートw0～w3およびr0ないしr3の選択により決定される。このビット記憶装置95－0～95－31の書込アクセスゲートw0～w3にはそれぞれ同一のアクセスゲート選択信号w0～w3（信号線とその上に伝達される信号とを同一の符号で示す）が伝達され、また読出アクセスゲートr0～r3には同一の読出ゲート選択信号r0～r3が伝達される。これにより、命令またはアドレスの32ビットが同時に選択される。

有効性フラグ記憶装置94はこのビット記憶装置95－0～95－31と同一の構成を有している。

第26図はビット記憶装置および有効性フラグ記憶装置の具体的構成の一例を示す図である。第26図において、ビット記憶装置（および有効性フラグ記憶装置）は、書込ポート選択トランジスタGT1～GT4と、読出ポート選択トランジスタRT1～RT4と、1ビットのデータを記憶する記憶素子MEを含む。記憶素子MEは、反並行または交差結合された2つのインバータIV1およびIV2を含む。すなわち記憶素子MEはインバータラッチにより構成される。

ゲートトランジスタGT1はポート選択信号w0をそのゲートに受け、メモリMEをポートwb0に接続する。ゲートトランジスタGT2は、ポート選択信号w1に応答して記憶素子MEをポートwb1に接続する。ゲートトランジスタGT3は、ポート選択信号w2に応答し記憶素子MEをポートwb2に接続する。ゲートトランジスタGT4はポート選択信号w3に応答して記憶素子MEをポートwb3に接続する。

読出経路も同様であり、トランジスタRT1～RT4はそれぞれ読出ポート選択信号r0～r3に応答して記憶素子MEをポートrb0～rb3へそれぞれ接続する。

このビット記憶装置および有効性フラグ記憶装置の異なるエントリ間の接続形態を第27図に示す。第27図に示すように、このポートwb0～wb3およびrb0～rb3はそれぞれ同一ビットの記憶装置に対して共通に設けられており、また有効性フラグ記憶装置においてはすべての記憶装置に対して共通に設けられる。

ビット記憶装置（有効性フラグ記憶装置）の動作について説明する。書込ワード選択信号WRDにより1本の書込ワード線が選択された場合、この第26図の構成において1個のゲートトランジスタGTiがオン状態となり、記憶素子MEがポートwb0～wb3のいずれかに接続される。これにより、選択されたポートを介して記憶素子MEへのデータの書込が行なわれる。データ読出も同様であり、読出デコーダRDDにより1個の読出ワード線が選択され、応じてトランジスタRT1～RT4のいずれかがオン状態となり、記憶素子MEがポートrb0～rb3のいずれかに接続される。これによりデータの読出が行なわれる。

第27図に示すように、1本の書込ワード線WDiを選択状態とすることにより、隣接するエントリ記憶装置においては異なるポートが選択される。たとえば第27図において記憶素子ME1は選択信号WDiによりポートwb0に接続され、記憶装置ME2がポートwb1に接続される。これにより、4つの異なる命令の各ビットを4つの異なる記憶素子へ同時に書込むことができる。データ読出も同様である。1本の読出ワード線RD1を選択状態とすることにより、記憶素子ME1，ME2，…が異なるポートへ接続され、同時に4つのエントリの内容がそれぞれ異なるポートに伝達されるため、同時に4つの命令、アドレスおよ

び有効性フラグを読出すことができる。

次に、このキューにおけるデータ読出時の動作について第２８図を参照して説明する。

この第２８図に示す動作波形図においては、命令読出ビット線ＩＲＢ、アドレス読出ビット線ＰＲＢおよび有効性フラグ読出ビット線ＡＲＢが読出ビット線ＲＢとして代表的に示されている。アドレスｓｃｏｐｅが与えられると、読出デコーダＲＤＤの出力は、このアドレスｓｃｏｐｅの変化状態に従って変化し、ある時間が経過するとその出力状態が確定し、１本の読出ワード線ＲＤが選択状態となる。この第２８図に示す動作波形図においては、すべての読出ワード線の信号波形図がすべて示されており、選択状態および非選択状態の読出ワード線の信号状態が示されている。この選択ワード線（たとえばＲＤ０）上の信号電位が確定すると、エントリ記憶装置の読出アクセスゲートが選択状態とされる。この場合、記憶装置９２－０～記憶装置９２－３のアクセスゲートｒ０、ｒ１、ｒ２およびｒ３がそれぞれ選択され、それぞれの異なるデータ読出ポートが選択状態とされ、対応のポート上にデータが読出される。この後、トランジスタＲＴ１～ＲＴ４により読出ビット線ＲＢ上の信号状態が確定する。この読出ビット線ＲＢ上の信号電位はＬラッチ回路１４～１６へ与えられる。

第２９図はデータ書込時の動作を示す信号波形図である。次にデータ書込動作について第２９図を参照して説明する。

アドレスｑｕｅｕｅ＿ｔｏｐが与えられると、書込デコーダＷＲＤは、このｑｕｅｕｅ＿ｔｏｐに従ってデコード動作を開始し、ある時間が経過した後、書込ワード線ＷＤ上の信号電位が確定状態となる。これにより、データ書込を受けるべき４つのエントリが選択され、かつこの４つのエントリ記憶装置のそれぞれの異なるポートがアクセスゲートを介して選択状態とされる。この書込ワード線ＷＤの信号電位が確定した後、キュー９に空領域があれば書込パルス発生器ＷＰＧより所定のタイミングで書込パルスＷＰが発生され、トライステートバッファＢＦ１～ＢＦ３が導通状態となり、命令ＩＣ＿ｄａｔａ、アドレスＩＦ＿ＰＣ＿Ｌおよび有効性フラグｉｎｓｔ＿ａｖａｉｌ＿ｉｎを書込ビット線ＷＢ上へ伝達する。この書込ビット線上のデータはそれぞれ選択されたエントリ（４つ）へそれぞれ同時に書込まれる。

書込んだデータを同時に読出すためには、書込パルスＷＰが発生され、書込ビット線ＷＢ上のデータが確定状態となった後に読出されることになるが、この場合、書込ビット線上に書込まれたデータが再びこのエントリ記憶装置を介してＬラッチ回路１４～１６へ伝達されたあとにこれらのＬラッチ回路がラッチ動作を実行する構成とされる。

また第２６図および第２７図に示す記憶素子はインバータラッチを用いており、このままでは、書込データと読出データが反転状態となることが考えられる。これは、トライステートバッファＢＦ１～ＢＦ３をインバータ構成とすれば容易に対処することができ、またこのインバータは任意の箇所に設置することができる。

上述の構成により、アドレスｑｕｅｕｅ＿ｔｏｐから始まる４つのアドレス領域に同時にそれぞれ異なるデータ（アドレス、命令および有効性フラグ）を書込むことができ、またアドレスｓｃｏｐｅから始まる４つのアドレス領域に格納されたエントリの内容を同時に読出すことができる。

なお、上述の記憶装置においては、記憶素子としてインバータラッチ構成の記憶素子が用いられているが、この記憶素子の構成はどのようなものであってもよく、データを保持するものであればどのようなものであってもよい。

また、記憶装置の構成は上述のエントリ記憶装置の構成に限定されず、データ書込ポートおよび読出ポートが４つ設けられており、それぞれ隣接する４つのエントリ記憶装置の異なる書込／読出ポートが選択状態とされる構成であればどのような構成であってもよい。

さらに、上記実施例においては、命令およびアドレスはともに３２ビットの場合および同時に読出される命令が４つの場合について説明したが、

これらのアドレスおよび命令長および同時にキューにおいて書込／読出される命令の数は任意の数であってもよい。

［発明の効果］

以上のように、この発明によれば、デコードステージにおいて、命令デコーダ前段に、命令メモリからの命令と、該命令の有効／無効を示すフラグと、該命令のアドレスとを１エントリとして複数の命令を同時に格納するキューを設け、命令メモリから同時に読出される複数の命令、およびアドレスならびに該命令の有効／無効を示すフラグを同時にキューに格納し、命令の実行状態、フェッチ状態等に従ってこのキューへの書込／読出を制御するように構成したため、命令デコーダへ効率的に命令供給を行なうことができ、処理速度の速い並列処理装置を得ることができる。

４．図面の簡単な説明

第１図はこの発明の一実施例である並列処理装置の命令供給ステージの構成を示す図である。第２図は第１図に示すキューの概念的構成を示す図である。第３図はこの発明による並列処理装置におけるキューのアドレスの基本的な動きを示す図である。第４図はこの発明の並列処理装置におけるキューの書込および読出アドレスおよび有効性フラグの動きを示す図である。第５図はこの発明による並列処理装置において命令フェッチ要求が出された場合に命令供給が行なわれなかった場合のキューにおける書込および読出アドレスと有効性フラグの動きを示す図である。第６図は第１図に示すキュートップロジックの実現する論理を一覧にして示す図である。第７図は第１図に示すスコープネクストロジックの実現する論理を一覧にして示す図である。第８図は第１図に示す命令フェッチロジックの実現する論理を一覧にして示す図である。第９図は第１図に示す命令ミスロジックが実現する論理を一覧にして示す図である。第１０Ａ図は第１図に示すキューアベールロジックの実現する論理を一覧にして示す図である。第１０Ｂ図は第１０Ａ図に含まれるキュー・フルの実現する論理動作を一覧にして示す図である。第１０Ｃ図および第１０Ｄ図は第１０Ｂ図に示すキュー・フルの論理動作を説明するための図である。第１１Ａ図および第１１Ｂ図はキューの第２の状態を説明するための図である。第１２Ａ図および第１２Ｂ図はキューの第３の状態を説明するための図である。第１３図は第１図に示すキューステイトロジックの実現する論理を一覧にして示す図である。第１４図はキューの第１図に示す命令アベールロジックが実現する論理を一覧にして示す図である。第１５図は第１図に示すキュー初期化状態ロジックが実現する論理を一覧にして示す図である。第１６図は第１図に示すキューが基本的動きを行なう際の各ロジックの動作を示す信号波形図である。第１７図は分岐発生時における第１図に示す各ロジックの動作を示す信号波形図である。第１８図は命令フェッチ要求に対し命令供給が行なわれなかった場合の第１図に示す各ロジックの動作を示す信号波形図である。第１９図は命令メモリからキューへ与えられる命令およびアドレスならびに有効性フラグの対応関係を概念的に示す図である。第２０図はキューへ与えられる命令の配置形態の一例を示す図である。第２１図はキューへ与えられる有効アドレスの配置形態の一例を示す図である。第２２図はキューへ与えられる有効性フラグの配置形態の一例を示す図である。第２３図はキューの全体の構成を示す図である。第２４図は第２３図に示すエントリ記憶装置の構成を示す図である。第２５図は第２４図に示す命令記憶装置およびアドレス記憶装置の構成を示す図である。第２６図は第２５図に示す命令およびアドレス記憶装置および有効性フラグ記憶装置の構成の一例を示す図である。第２７図は第２６図に示すビット記憶装置の接続形態を例示する図である。第２８図はキューのデータ読出時の動作を示す信号波形図である。第２９図はキューのデータ書込時における動作を示す信号波形図である。第３０図は並列処理装置の概念的構成を示す図である。第３１図は、並列処理装置の一般的構成を示す図である。第３２図は従来の並列処理装置における命令供給方法を示す図である。第３３図は、

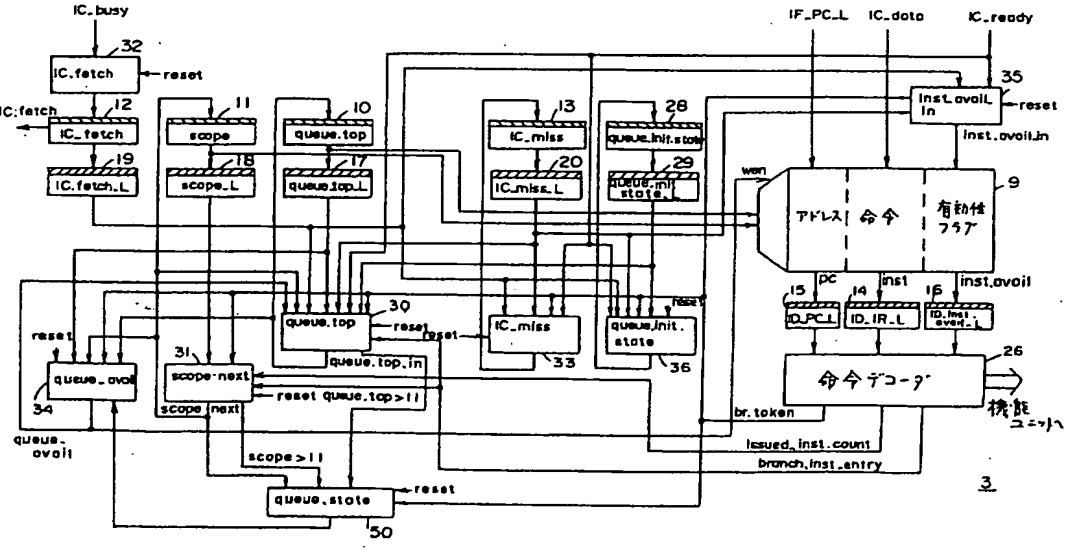
この発明が意図する命令供給方式を示す図である。第３４図は、第３３図に示す命令供給方式を実現するために考えられることのできる命令供給方式を示す図である。第３５図は第３４図の命令供給方式を実現するための構成の一例を示す図である。第３６図は並列処理装置において用いられる２相クロックを示す図である。

図において、１は命令メモリ、２は命令フェッチステージ、３は命令デコードステージ、９はキュー、３０はキュートップロジック、３１はスコープネクストロジック、３２は命令フェッチロジック、３３は命令ミスロジック、３４はキューアベールロジック、３５は命令アベールロジック、３６はキュー初期化状態ロジック、５０はキューステイトロジック、９２－０～９２－１１はエントリ記憶装置、９３－１は命令記憶装置、９３－２はアドレス記憶装置、９４は有効性フラグ記憶装置、９５－０～９５－３１はビット記憶装置、ＭＥは記憶素子、ＷＲＤは書込デコーダ、ＲＤＤは読出デコーダ、ＷＰＧは書込パルス発生器、ＢＦ１～ＢＦ３はトライステートバッファである。

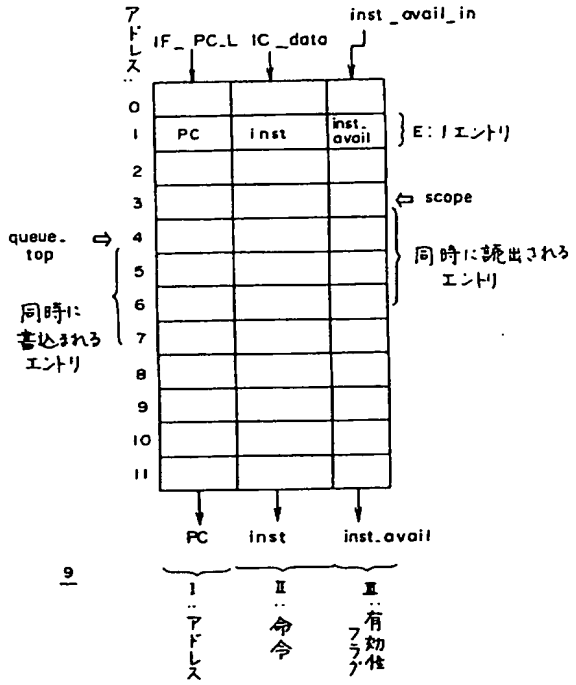
なお、図中、同一符号は同一または相当部分を示す。

特許出願人　三菱電機株式会社
代　理　人　弁理士　深　見　久　郎
（ほか２名）

第１図



第2図



第3図

キュー，queue_top，scope の基本的な動き

キューのアドレス	0	1	2	3	4	5	6	7	8	9	10	11
(1) 命令	x	x	x	x	x	x	x	x	x	x	x	x
有効性フラグ	0	0	0	0	x	x	x	x	x	x	x	x
queue_top	↓											
scope	↓											
(2) 命令	0	1	2	3	x	x	x	x	x	x	x	x
有効性フラグ	—	—	—	—	x	x	x	x	x	x	x	x
queue_top	↓											
scope	↓											
(3) 命令	0	1	2	3	4	5	6	7	x	x	x	x
有効性フラグ	—	—	—	—	—	—	—	—	x	x	x	x
queue_top					↓							
scope			↓									
(4) 命令	0	1	2	3	4	5	6	7	8	9	10	11
有効性フラグ	—	—	—	—	—	—	—	—	—	—	—	—
queue_top									↓			
scope				↓								
(5) 命令	0	1	2	3	4	5	6	7	8	9	10	11
有効性フラグ	—	—	—	—	—	—	—	—	—	—	—	—
queue_top	↓											
scope				↓								
(6) 命令	12	13	14	15	4	5	6	7	8	9	10	11
有効性フラグ	—	—	—	—	—	—	—	—	—	—	—	—
queue_top	↓											
scope							↓					

1サイクル

x：未定

第4図

分岐が生じたときのキュー，queue_top，scopeの動き

キューのアドレス 0 1 2 3 4 5 6 7 8 9 10 11

(11) 命令 10 11 0 1 2 3 4 5 6 7 8 9
有効性フラグ 1 1 1 1 1 1 1 1 — — — —
queue_top —
scope —

(12) 命令 10 11 0 1 2 3 4 5 6 7 8 9 ：命令2が分岐命令で分岐が生じた場合
有効性フラグ 1 1 1 0 0 0 0 1 1 1 — —
queue_top —
scope —

(13) 命令 10 11 0 1 20 21 22 23 6 7 8 9 ：分岐先命令は20
有効性フラグ 1 1 1 1 1 1 1 1 1 1 — —
queue_top —
scope —

第5図 プロセッサからの命令フェッチ要求に対して，命令メモリから命令を供給できない場合のキュー queue_top，scopeの動き

キューのアドレス 0 1 2 3 4 5 6 7 8 9 10 11

(21) 命令 8 9 10 11 0 1 2 3 4 5 6 7
有効性フラグ 1 1 1 1 1 1 1 1 1 1 1 1
queue_top —
scope —

(22) 命令 8 9 10 11 0 1 2 3 4 5 6 7 ：命令が命令メモリから供給されなかった
有効性フラグ 1 1 1 1 0 0 0 0 1 1 1 1
queue_top —
scope —

(23) 命令 8 9 10 11 0 1 2 3 4 5 6 7 ：命令供給待ち
有効性フラグ 1 1 1 1 0 0 0 0 1 1 1 1
queue_top —
scope —

(24) 命令 8 9 10 11 12 13 14 15 4 5 6 7 ：命令到着
有効性フラグ 1 1 1 1 1 1 1 1 1 1 1 1
queue_top —
scope —

第6図

queue_top の論理

reset	queue_init_state_L	br_taken_L	IC_fetch_L	IC_miss	IC_ready	queue_avail	queue_top_in
1	x	x	x	x	x	x	0
x	1	x	x	x	x	x	queue_top_L
x	0	1	x	x	x	x	branch_inst_entry(※1)
0	x	x	x	x	x	x	queue_top_L
0	0	0	1	x	0	x	queue_top_L
0	0	0	x	1	x	1	queue_top_L
0	0	0	0	x	1	0	queue_top_L
0	0	0	x	1	1	1	(※2)
0	0	0	0	x	1	0	queue_top_L
0	0	0	0	0	x	x	queue_top_L
0	0	0	0	0	0	0	queue_top_L

x：任意

(※1) branch_inst_entry＝命令キャッシュで検索されたブランチ命令のターゲットアドレス：queue_state＝NORMAL（queue＝第2の状態）

(※2) queue_top_in＝queue_top_L＋4；
if (queue_top_in>11)
queue_top_in=queue_top_in－12；
queue_state＝REVERSE：if scope>11＝0；else queue_state＝NORMAL）

第7図 scope_next の論理

(1) reset : scope_next = 0 ;

(2) br_taken: scope_next=branch_inst_entry;

(3) else: scope_next = scope_L + 発行命令数(issued_inst_count);
if (scope_next > 11)
scope_next = scope_next - 12 ;

第8図 IC_fetch の論理

(1) /reset and /IC_busy: IC_fetch = 1;
(2) else. IC_fetch = 0;

第9図 IC_miss の論理

reset	br_taken	IC_fetch_L	IC_ready	IC_miss
1	x	x	x	0
0	1	x	x	0
0	0	1	0	1
0	0	x	1	0
0	0	0	0	IC_miss_L

x：任意

第10A図

queue_avail の論理

reset	br_taken	queue_full	queue_avail
1	x	x	1
0	1	x	1
0	0	0	1
0	0	1	0

x：任意

第10B図

queue_full

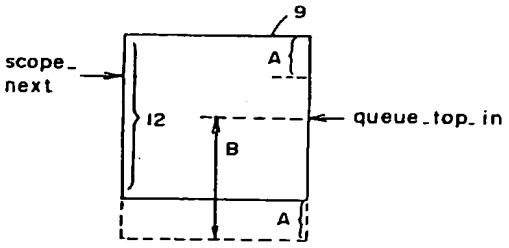
scope_next : ※サイクルの scope;
queue_top_in : ※サイクルの queue_top;

(1) scope_next < queue_top_in :
if (scope_next + 12 - queue_top_in >= 4) FALSE;
else TRUE;

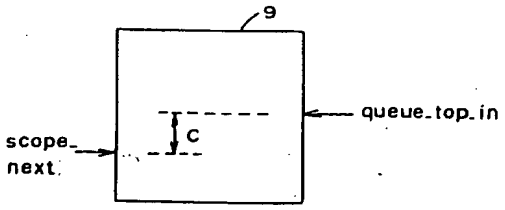
(2) scope_next > queue_top_in :
if (scope_next - queue_top_in >= 4) FALSE;
else TRUE;

(3) scope_next == queue_top_in :
if (queue_state == NORMAL) FALSE
else TRUE;

第10C図



第10D図



第11A図

NORMAL状態

キュー	命令	ポインタ
0		
1		
2	x	scope
3	x	
4	x	
5	x	
6		queue_top
7		
8		
9		
10		
11		

x：未発行命令

第11B図

NORMAL状態で scope_next queue_top_in

キュー	命令	ポインタ
0		
1		
2		
3		
4		
5		
6		scope_next queue_top_in
7		
8		
9		
10		
11		

第12A図

REVERSE状態

キュー	命令	ポインタ
0	x	
1	x	
2		queue_top
3		
4		
5		
6	x	scope
7	x	
8	x	
9	x	
10	x	
11	x	

：x：未発行命令

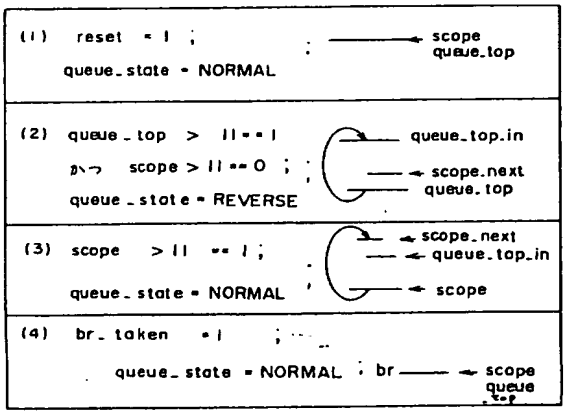
第12B図

REVERSE状態で scop_next:queue_top_in

キュー	命令	ポインタ
0	x	
1	x	
2	x	
3	x	
4	x	
5	x	
6	x	scope_next queue_top_in
7	x	
8	x	
9	x	
10	x	
11	x	

：x：未発行命令

第13図



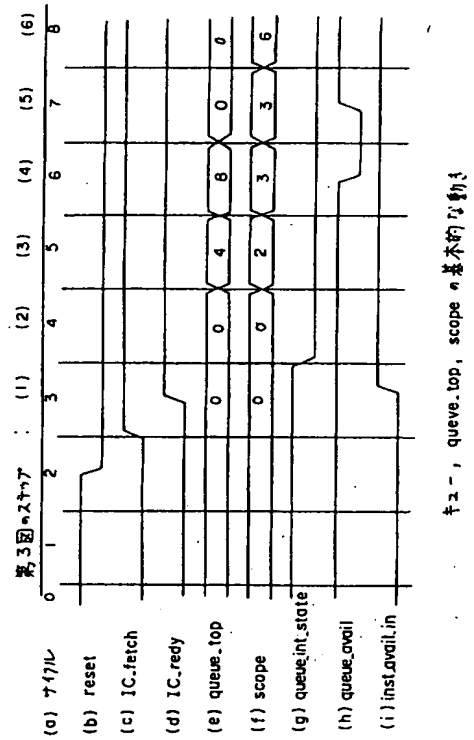
第14図

キューの「有効性フラグ」部分に書き込む値の論理

reset	br_taken	IC_fetch_L	IC_miss_L	inst_avail_in
1	x	x	x	0
0	1	x	x	0
0	0	1	x	IC_ready
0	0	x	1	IC_ready
0	0	0	0	0

x：任意

第16図



第15図

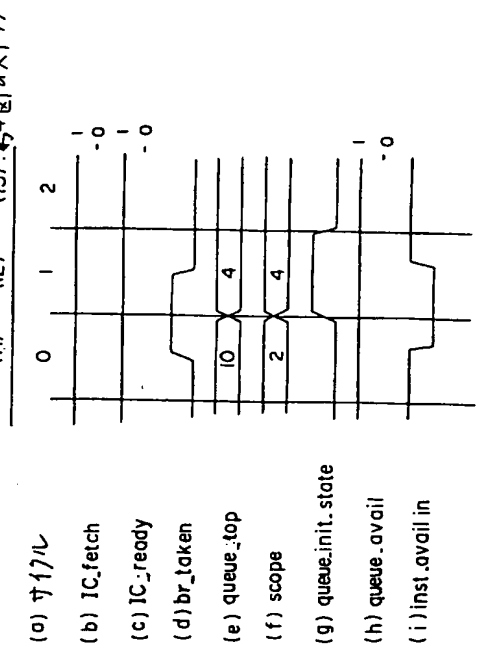
queue_init_state の論理
(ロジック36)

reset	br_taken	IC_fetch_L	IC_miss_L	IC_ready	queue_init_state	queue_init_state_L
1	x	x	x	x	1	0
x	1	x	x	x	1	0
0	0	1	x	0	1	0
0	0	0	x	1	0	1
0	0	0	1	1	0	1
0	0	0	0	x	1	0

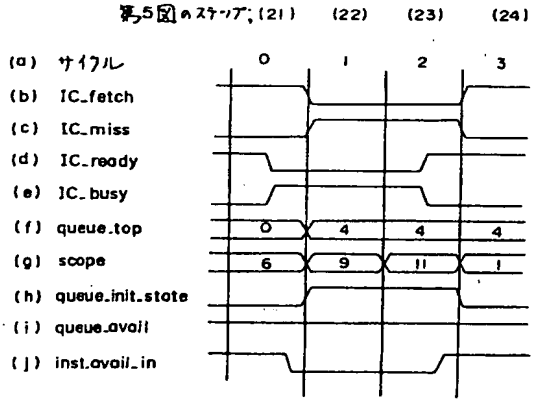
x: 任意

第17図

分岐発生時のキュー, queue_top, scope の動き

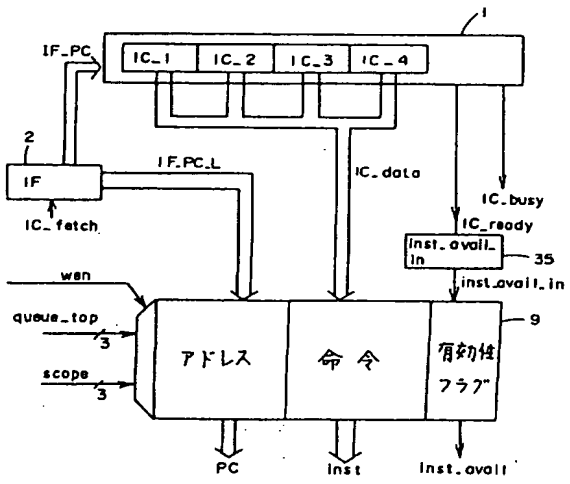


第18図

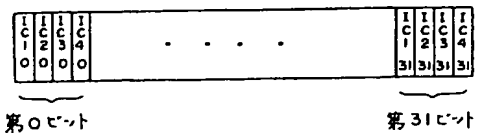


: 発生された命令フェッチ要求に対して命令メモリから命令が供給されない場合のキュー, queue_top および scope の動き

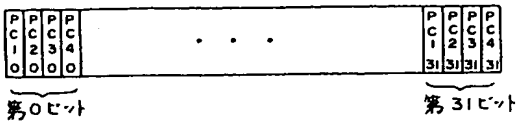
第19図



第 20 図



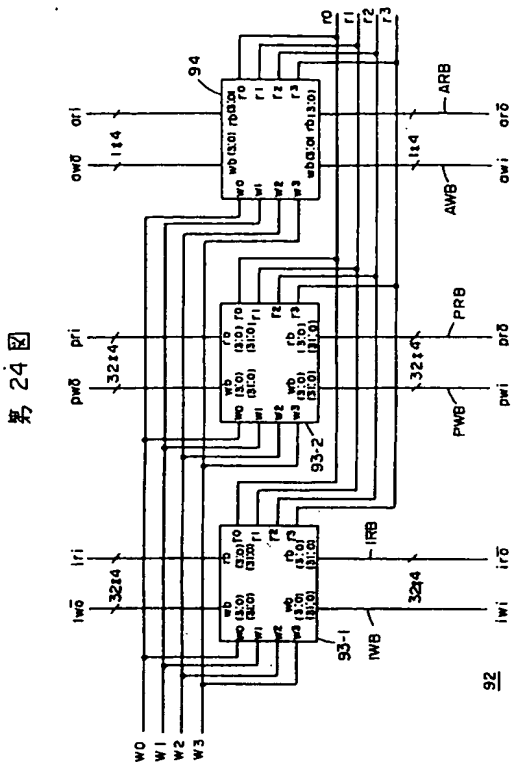
第 21 図



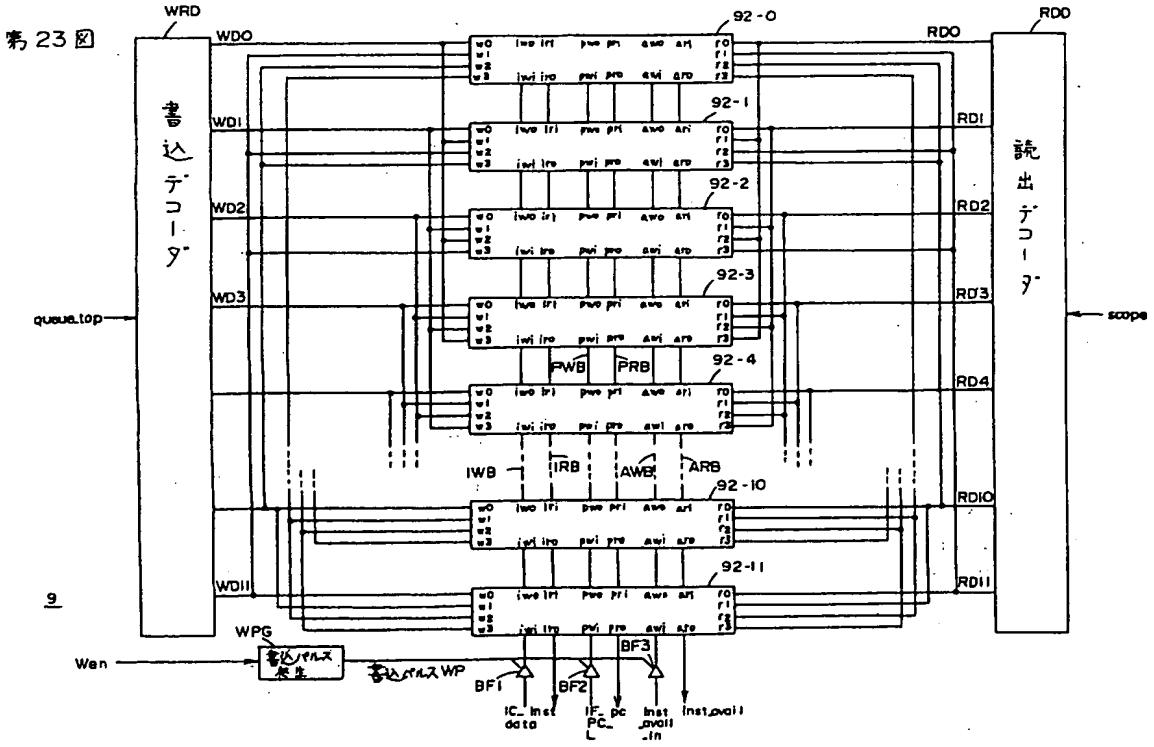
第 22 図

inst_avail_in_0	inst_avail_in_1	inst_avail_in_2	inst_avail_in_3

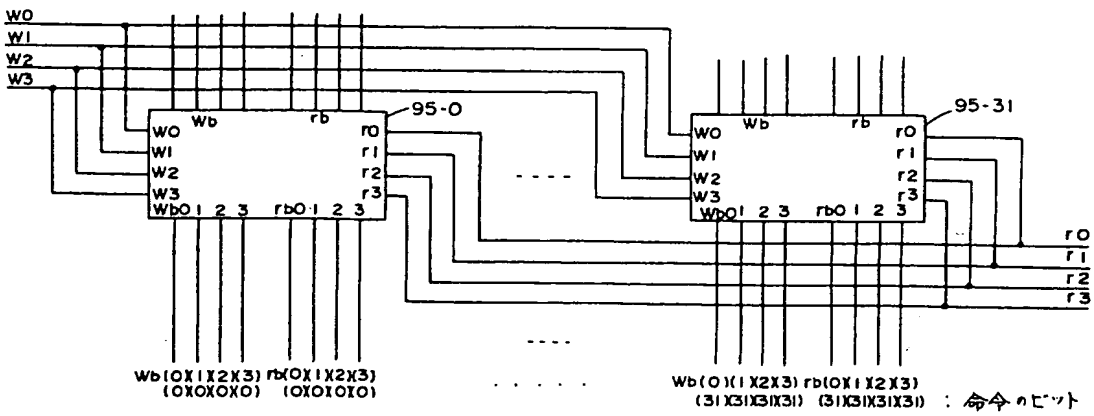
第 24 図



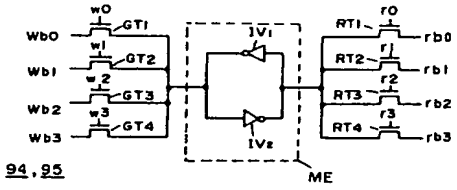
第 23 図



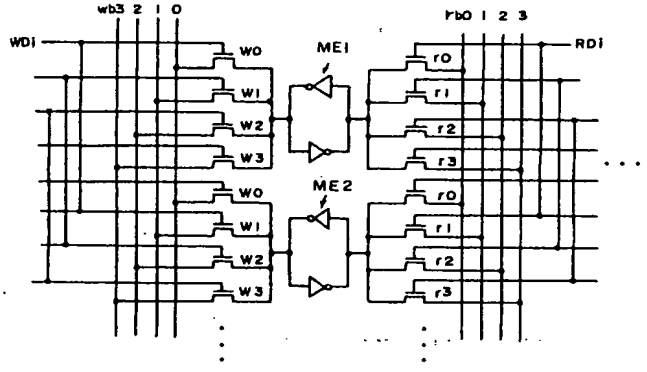
第 25 図



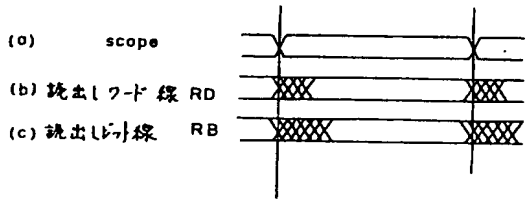
第 26 図



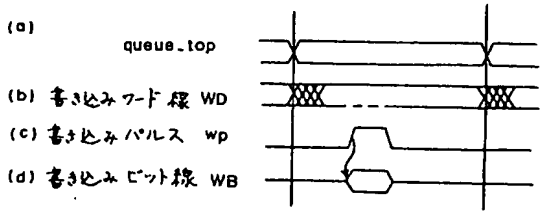
第 27 図



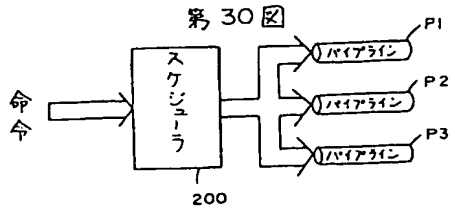
第28図



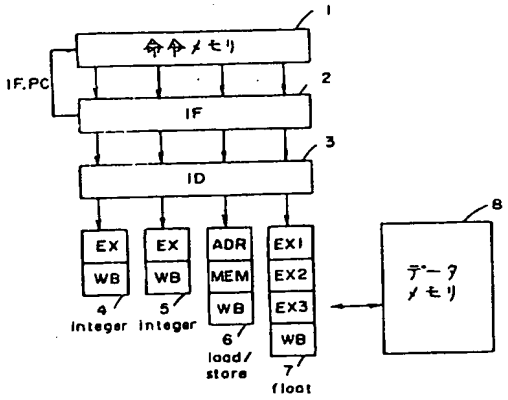
第 29 図



第30図



第31図



第32図

サイクル	命令	
1	[1] 2 3 4	; 1の発行
2	[2] [3] 4	; 2,3の発行
3	[4]	; 4の発行
4	[5] [6] 7 8	; 5,6の発行
5	[7] [8]	; 7,8の発行
6	...	

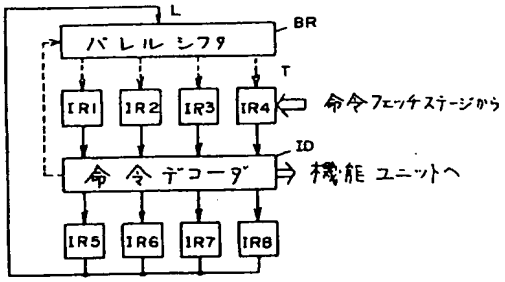
第33図

サイクル	命令				
1	[1]	2	3	4	; 1の発行
2	[2]	[3]	4	5	; 2,3の発行
3	[4]	[5]	[6]	7	; 4,5,6の発行
4	[7]	[8]	9	10	; 7,8の発行
5		...			

第34図

(1) [2] [3] 4 5 ; 命令2,3の発行
(2) 4 5 — — ; 命令レジスタのシフト
(3) 4 5 6 7 ; 空いた命令レジスタへの命令のフェッチ

第35図



第36図